OPC技術概要書

(Data Access Ver2および共通仕様)

Version2.0

1999年11月

日本OPC協議会技術部会

改訂版発行にあたり

1997年2月にOPC技術概要書（第1版）を発行してから約2年が経過しました。この間，OPC Foundation（OPC-F）ではData Access1.0A（DA1.0A）を1997年5月に一般公開し，またDA2.0も1998年11月上旬に一般公開されました。こういった状況から，OPC技術概要書もDA2.0の最新仕様を取り込むべく改訂の必要が生じたため，日本OPC協議会（OPC-J）技術部会メンバーが，第1版同様改訂版の執筆を担当し，ここに改訂版発行の運びとすることができました。これもひとえに技術部会メンバーの努力の賜物であり，それを支えていただいた幹事会メンバーおよび会員皆様方のご協力なくしては成し遂げることはできなかったことと思います。改めてお礼申し上げます。

さて，DA1.0Aは，1996年8月に一般公開されたDA1.0のマイナーチェンジでありましたが，DA2.0ではオートメーションインターフェースの前面改訂やオートメーションラッパーの提供など大きな仕様変更となっています。さらに，OPC-Fでは，Alarm&Event(A&E)やHistorical Data Access（HDA）の策定もほぼ完了となり，順次一般公開されることになっています。OPC-J技術部会では，引き続きA&EやHDAに関する技術概要書を発刊する予定であり，OPC-J会員各位のOPC技術習得に役立ち，ひいてはOPC技術の普及に寄与したいと考えております。今後ともOPC会員各位のご協力をよろしくお願いいたします。

本書は下記構成になっています。

第1章　OPC概要：OPC全体の概要

第2章　OPC機能：OPC全体の機能とOPC仕様区分について

第3章　OPC Data Access概要：OPCの仕様の一つであり、基本部であるData Access の概要について

第4章　OPC 構造と動作；、OPCサーバー・クライアントの構造とData Access仕様によるデータの読み取り、書き込み動作について

第5章　OPC対応ソフトウェア開発：OPC仕様のサーバ・クライアントの開発に必要な基本事項について

第6章　OPCアプリケーションとOPCインタフェース：OPCサーバー・クライアントの応用例とData Access 仕様におけるインタフェースの機能一覧

第7章　DataAccess Ver1.0とVer2．0の主な違い：DataAcessのバージョンによる違いについて

第8章　OPC共通仕様：OPC個別仕様に共通している特記事項について

第9章　技術基盤説明：基盤技術概要

第10章 付録：用語の解説

本書では、単にOPCと記したばあいには、Data Access仕様のみならず、他の仕様をも包含した意味で使用しています。OPC Data Access を他の仕様と区別して記述する必要のある場合には、Data Accesまたは単にDAと省略して記載します。

なお本文中アンダーラインを引いた語（各章の初出のみ）は技術基盤 あるいは付録の用語集などの章で簡単に説明されています。

目次

1 OPC 概要

1.1 背景

以前のコンピュータシステムではハードウェアやソフトウェアなどプラットホームが異なるコンピュータ間のデータ共有、通信をおこなうために独自にプログラムを作成する必要があり、多大な労力を費やしてきましたが、現在は確立した規格化のおかげで、たとえばインターネットのように、異なるコンピュータ間を接続する広大なネットワークが実現されました。このため、例えば企業情報システムの開発においては規格に基づいたデータベースとクライアント／サーバツールを使って、本来の目的であるアプリケーション開発に注力できるようになりました。製造システムにおいても同様な進化が必要です。すなわち、異なるベンダの機器を特別なソフトウェアを開発することなく相互接続できること、また、図 1-1のようなシステムを構築する上で、プラント機器レベルのデータを扱うフィールド管理層、プロセス処理を行う分散制御システム層、さらに上位の生産管理層に至るまでシームレスにデータ共有化がはかれる規格作りと普及が急務です。このような状況のもとで製造システムにおいても今後の主流とみられるWindowsの基幹技術であるOLE/COMをプロセス制御用途に拡張するOPCが策定されました。

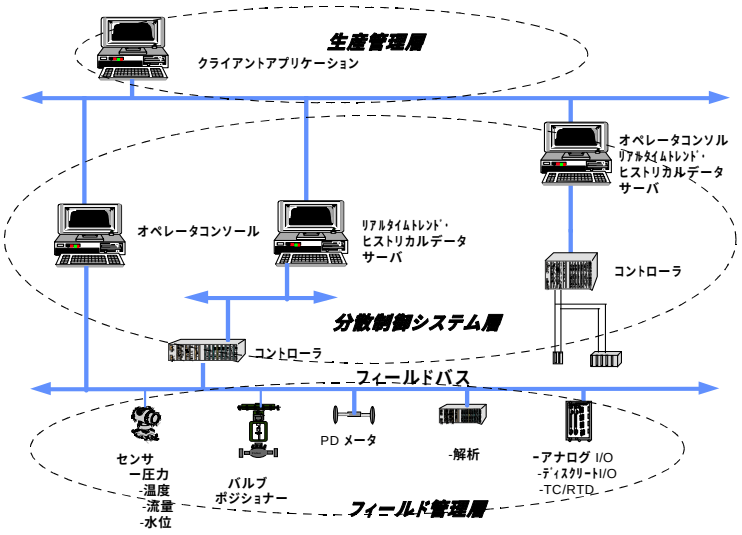


図 1-1 プロセス制御情報アーキテクチャ

1.2 OPCの特長

1.2.1 目的とスコープ

OPCは異なるベンダの機器で実行される異なるアプリケーション間のインタフェースを標準化してデータ交換を容易にすることが目的です。その結果として、開発言語や開発環境などにかかわらず統合的に利用できるプロセス制御用ソフトウェアコンポーネントをユーザへ提供できるようになります。また、現状ハードウェアのドライバインタフェースとこれらを利用するアプリケーション間のインタフェースには統一性がないため、装置ごとあるいはベンダごとに異なり、たとえば図 1-2の様に3種類のサーバ（装置）とそれを利用するアプリケーションが2種類ある時、合計6種類のインタフェースソフトを多大な時間をかけて作成する必要があります。OPCを利用することによりインタフェースを1種類に統一することができ、図1-3のようなシステムを構築することができます。アプリケーションX（Y）からは装置A、B、Cの内部仕様、ベンダなどの変更に無関係に装置を利用することができます。

OPCを利用するシステムは、大きく、アプリケーション（クライアント）の要求に応じてデータ収集ほかサービスを提供するOPCサーバ、OPCサーバを利用するためのOPCインタフェース、およびサービスをうけるOPCクライアントを用いたアプリケーションで構築することができます。OPCサーバは各ベンダのハードウェアに実装することができますが、OPCサーバにはベンダごとのハードウェア、システムの違いをベンダ固有機能対応として処理する機能、およびデータのフォーマットもクライアントの要求に応じて対応できるVARIANT型とよばれるデータタイプを処理する機能があるからです。

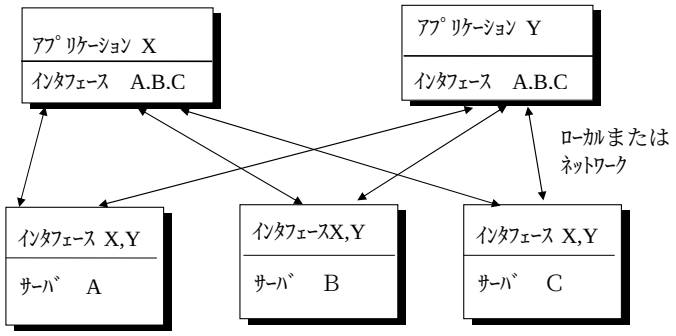


図 1-2 サーバとアプリケーション

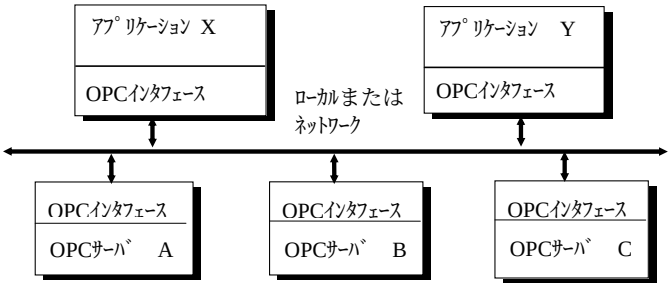


図 1-3 OPCサーバとアプリケーション

また、OPCは(1)移植しやすく、(2)多くのベンダのニーズに応えられるようフレキシブルで、(3)ハイレベルな機能性を提供し、(4)十分なオペレーションが可能であることを目標に策定されており、メーカやユーザにとってそれぞれ次のようなメリットをもたらします。

(1)装置開発者:デバイスドライバ開発の一元化が可能になります。

(2)アプリケーションソフトウェア開発者:汎用開発ツールを利用できます。独自インタフェースの開発が不要になり、装置とのインタフェースが容易になります。

(3)ユーザ:各種業務パッケージソフトとの連携が可能になり、システム開発コストを低減できます。また、フィールドデバイスやI/O機器を統合し、表示、操作性の統一されたシステムを構築できます。

OPC仕様準拠のコンポーネントが普及するにつれシステムの拡張やコンポーネントの置換が容易になるほか、データアクセスも容易になり、たとえばプロセス制御アプリケーションでのデータをデータベースソフトや表計算ソフトに直接送信し自動編集するなど高度なデータ統合が可能になります。

なお、OPC仕様を実装したサーバの実行環境はWindows NT 4.0以降を必要とします。OPCクライアントの実行環境はWindows 95(クライアントのみ実行する場合)またはサーバ同様Windows NT 4.0以降を必要とします。 機能的には装置間の効率的データ読み込み／書き込みに限定されています。アラーム、セキュリティ、履歴管理などは別仕様で追加される予定です。

1.2.2 システムにおける位置づけ

OPCはデータの供給元とデータの利用者間を接続するためのインタフェース仕様です。データの供給元にはたとえばPLC、DCS、バーコードリーダ、計測解析機などが考えられます。図 1-4の例のように、最も低いレベルとしては物理デバイスからの生データをDCS／SCADA、自動化制御システムへの供給のために、また、さらにそこからのデータを上位アプリケーションに送る用途や、直接アプリケーションと物理デバイスとを接続する場合などにも適用できます。このようにOPCインタフェースはシステム上多くの場所に数多く実装することができる柔軟な仕様になっています。

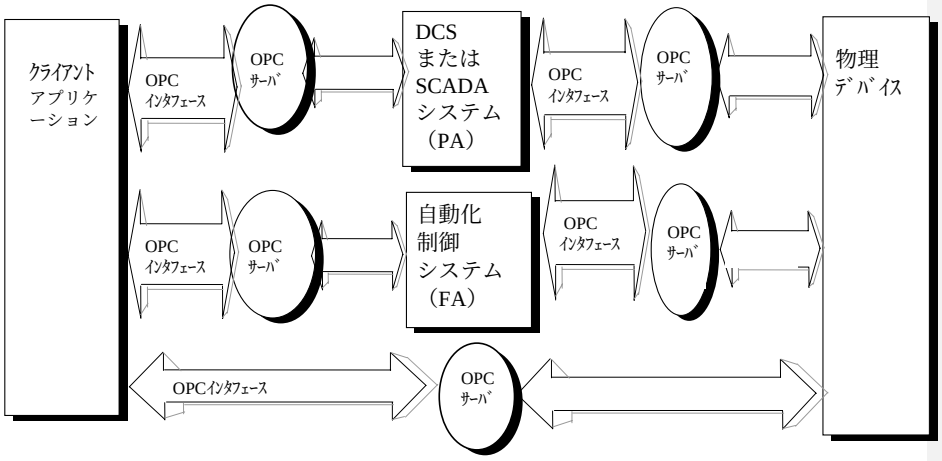


図 1-4 OPCとシステムにおける位置づけ

1.2.3 開発言語とOPCの実装形態

OPCはマイクロソフトのOLE/COM技術に基づき、クライアント／サーバモデルを継承しています。OPCサーバの実装形態には、クライアントと同一マシン上でクライアントと同じプロセスで実行するモデル（In-Proc Server）、別のプロセスで実行するモデル(Local Server)、さらにネットワーク上の他のマシン上で実行するモデル(Remote Server)などがありますが、ネットワーク構成の場合を例に図1-5に実装形態を示します。

OPCサーバはDCSやI/Oドライバなどを供給するハードウェアベンダが提供するソフトウェアで、図1-5では右側のサーバマシン上で実行されます。そのクライアントは左側のクライアントマシン上で実行されるIn-Proc Handlerです。ユーザインタフェースなどを提供するアプリケーションベンダは左側のクライアントアプリケーションプログラムを開発します。開発言語にはVB(Visual Basic)やC++などを利用することができます。一般には、VBや、VBA（VB for Applications）のようなスクリプト言語で開発されるアプリケーションはOPCオートメーションインタフェースを利用し、アプリケーションがC++で作成され、また、最大性能を引き出したい場合はOPCカスタムインタフェースを利用します。クライアントアプリケーションでOPCオートメーションインターフェースを利用する場合には、Automation Wrapper ｲﾝﾀｰﾌｪｰｽを使用することでOPCカスタムインターフェースに変換され、サーバーにアクセスすることができます。

OPCクライアントアプリケーションから、たとえばプロセス制御装置などへの要求はIn-Proc Handlerを経由してネットワーク上にあるサーバマシンのOPCオブジェクトのインタフェースを介して伝わります。具体的にはメソッド、プロパティをアクセスしてデバイスへのデータ設定、デバイスからの読みとりを行います。OPCサーバはこれらの呼びかけで要求に応じた適切な対応（たとえば、装置のﾃﾞｰﾀなど）をOPC仕様に準拠して用意しなければなりません。この装置固有の対応付けなどはOPCサーバを供給するベンダの責任で行います。

In-Proc Handlerはサーバ／クライアント間のデータをバッファリングしておく一種のデータキャッシュで、OPC仕様上必須ではありませんが、ネットワーク間の通信時間を節約して高い性能を実現するために、OPCサーバとともにハードウェアベンダが提供することを推奨されているプログラムです。

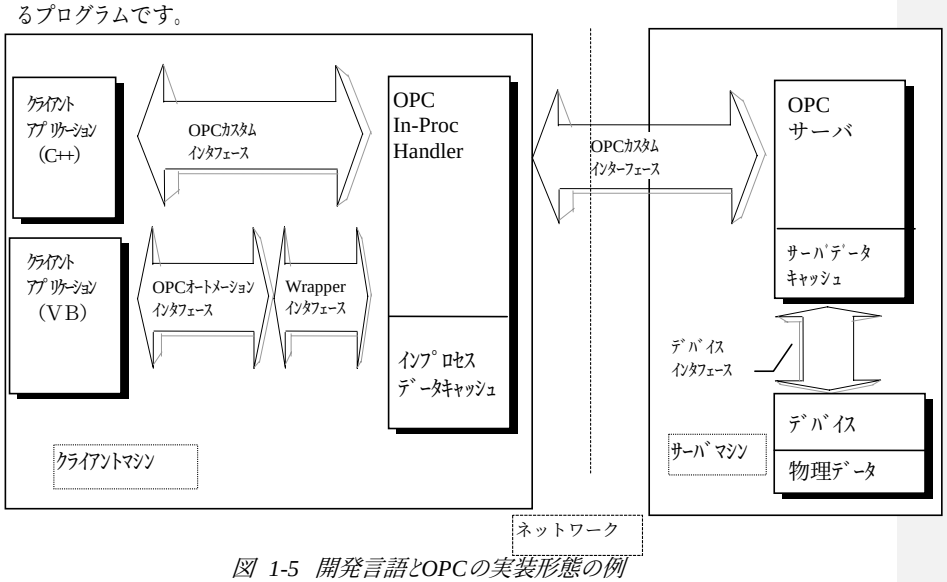


図 1-5 開発言語とOPCの実装形態の例

2 OPC 機能

2．1概要

OPCはﾘｱﾙﾀｲﾑにﾌﾟﾛｾｽﾃﾞｰﾀを転送する機能を実現するための拡張性を考慮したﾌﾚｰﾑﾜｰｸを提供します。ｱﾌﾟﾘｹｰｼｮﾝはﾌﾟﾛｾｽﾃﾞｰﾀがどのようなﾌｫｰﾏｯﾄで、どこにあり、どのように取得しなければならないかを意識することなくﾌﾟﾛｾｽﾃﾞｰﾀｱｸｾｽ（読み込み/書き込み）ができます。

OPCはｸﾗｲｱﾝﾄ/ｻｰﾊﾞ構造をとっています。1つのOPCｸﾗｲｱﾝﾄは1つ以上のﾍﾞﾝﾀﾞが提供するOPCｻｰﾊﾞ（複数）に接続できます。逆に、1つのOPCｻｰﾊﾞは複数のOPCｸﾗｲｱﾝﾄとの接続をｻﾎﾟｰﾄします。

OPCｻｰﾊﾞは物理ﾃﾞﾊﾞｲｽやそれに対するｱｸｾｽ方法を知っており、OPCｸﾗｲｱﾝﾄはそれらについて知っている必要はありません。

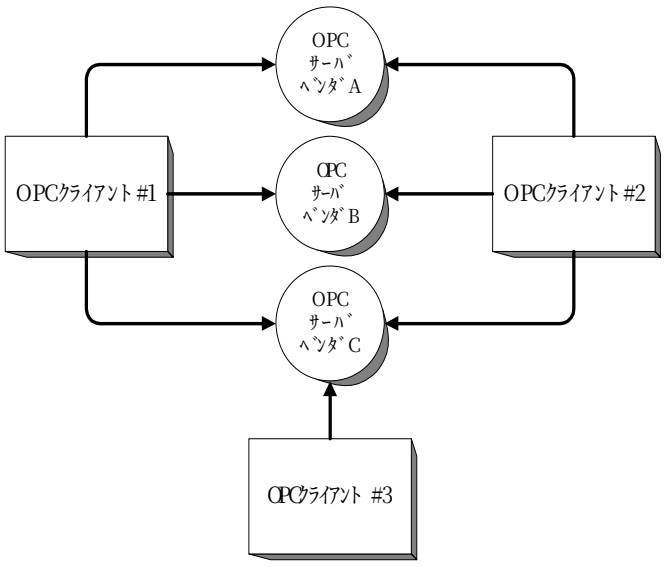


図 2-1 OPCのｸﾗｲｱﾝﾄ／ｻｰﾊﾞ（例）

OPCはﾃﾞｰﾀ交換に関するｲﾝﾀﾌｪｰｽ規約 であり、この章ではOPCにより実現される機能について、FA／PAのｼｽﾃﾑをﾓﾃﾞﾙとして説明します。

一般に、監視･制御ｼｽﾃﾑではﾌｨｰﾙﾄﾞ機器に直結した現場ｺﾝﾋﾟｭｰﾀが機器からのﾃﾞｰﾀ収集、保管、監視、制御、診断の各機能を担当します。

従来、ﾃﾞｰﾀの収集、保管、監視、制御、診断、伝送、ｱﾗｰﾑ表示、その他の業務処理のための機能ﾓｼﾞｭｰﾙはﾒｰｶｰなどが独自に設計し、そのｲﾝﾀﾌｪｰｽを規定しています。

これらの機能ﾓｼﾞｭｰﾙを標準の論理ｵﾌﾞｼﾞｪｸﾄとして整理し、その呼び出しｲﾝﾀﾌｪｰｽを標準化することにより、FA/PAｼｽﾃﾑのｿﾌﾄｳｪｱ構築を次のように進めることがOPC適用のおもなねらいです。

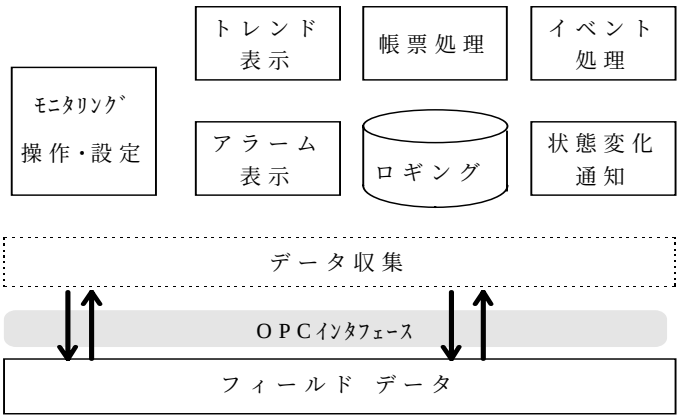


図 2-2 監視･制御システムの機能構成例

監視･制御システムでは、ﾌﾟﾗﾝﾄ、瞬時データ、ﾄﾚﾝﾄﾞ、アラームをはじめ、各種業務処理用の画面が存在します。このような表示画面をオブジェクト部品として整備していけば、それらを利用することによってシステムの生産性を大きく向上できます。さらに、ユーザ仕様に不確定部分の残る初期段階でのプロトタイプ作成や仕様変更に容易に対応できます。

機能部品を実現するためにｸﾗｲｱﾝﾄとしては、①ﾌｨｰﾙﾄﾞﾃﾞｰﾀ収集②ﾋｽﾄﾘ③ﾚﾎﾟｰﾄ④アラーム／ｲﾍﾞﾝﾄなどの各データを必要とします。

現状ではデータ収集を受け持つｱﾌﾟﾘｹｰｼｮﾝがデータのロギングを行い、また、データの整合性等をﾁｪｯｸし異常を検出した場合にはアラームの通知を行う等の処理を実装します。したがって、アラームの表示方法等についてはデータ収集などを行うｱﾌﾟﾘｹｰｼｮﾝの処理に依存します。

2．2 仕様の種類

現状 OPC仕様として、下記4種類の種類があります。

表2．1 OPC仕様の種類

仕様	ﾊﾞｰｼﾞｮﾝ	内容
OPC Data Access	1．0A、2．0	データのアクセスの規定
OPC Common	1．0	OPC仕様の共通部分
Alarm and Events	1．0	アラーム･ｲﾍﾞﾝﾄの規定
Historical Data Access	1．0	ﾋｽﾄﾘデータアクセスの規定

：本概要書記載範囲

本概要書では、ﾌｨｰﾙﾄﾞﾃﾞｰﾀのアクセス方法について規定しているOPC DataAccess 仕様とOPCの共通仕様であるOPC Commonに関して概要を記載するものです。ﾋｽﾄﾘ、アラーム／ｲﾍﾞﾝﾄなどについて規定している仕様として、OPC Historical Data Access ならびに OPC Alarm a

nd Eventsがあります。Historical Data Access ならびに Alarm and Eventsに関しては、別冊として概要書を作成する予定です。

3 OPC Data Access 概要

3．1機能概要

OPC Data Access（以下 DAと記載）を実装することにより実現される機能を以下に示します。DAはこれらをすべて必須として要求しているのではなく、オプションと定めているものがあります。その場合、これらが実際にサポートされるかどうかはベンダ依存、サーバ依存となります。オプションか否かは機能単位よりもむしろインタフェース単位で定められており、この章でも概略が示されていますが、詳細については4章またはOPC DAの仕様書 を参照してください。

表 3-1 DAの機能仕様

機能	方式	説明
プロセスデータ読み込み ※ 4つの方式は、 互いに独立していて 干渉はありません。	同期読み込み	アイテムに対応するプロセスデータを読み込みます。 読み込み完了まで待ち合わせます。
	非同期読み込み	アイテムに対応するプロセスデータを読み込みます。 要求後直ちにリターンし、読み込み動作が完了した時点でDAクライアント側のメソッドが呼び出されます。 要求時に受け取りデータのフォーマットを指定できます。 ・タイムスタンプ付きデータ ・タイムスタンプなしデータ
	リフレッシュ	すべてのアクティブなアイテムからプロセスデータを読み込みます。 （非同期入力の一種）
	データ変化通知 (subscription)	一定周期でチェックし、データにある幅以上の変化があった場合、DAクライアントに通知します。
プロセスデータ書き込み	同期書き込み	アイテムにデータを出力します。 書き込み動作完了まで待ち合わせます。
	非同期書き込み	アイテムにデータを出力します。 要求後直ちにリターンし、書き込み動作が完了した時点でDAクライアント側のメソッドが呼び出されます。
構成情報のセーブ／ロード	—	DAサーバ全体の構成情報のセーブ／ロードをサポートします。
ブラウズ	—	グループ、アイテム、属性などの各種情報を閲覧します。

DAはその機能仕様よりは実現の仕方、実装の仕方に特長があります。

実際には、これらの機能がDAカスタムインタフェースとDAオートメーションインタフェースという用途別の2つのインタフェースで提供されるよう規定されています。

これらの使い分けについては4.2節を参照してください。

3．2 プロセスデータアクセスの基本

基本的なアイデアはDAクライアントがDAサーバを指定して接続し、接続したDAサーバでアクセスできるアイテムの集合（サーバアドレススペースという）中の個々のアイテムを識別する名前（アイテムID）を指定して、それに対応するプロセスデータを読み込み/書き込みするというものです。実際には、効率向上のためグループという概念を設け、グループのなかに1つあるいは複数のアイテムを対応させます。

DAが定めたオブジェクトモデルでは標準コンポーネントとして、DAサーバオブジェクト、DAグループオブジェクト、DAアイテムオブジェクトと呼ぶ3種類の論理オブジェクトを提供しています。
DAサーバは図 3-1のようにベンダごとの物理デバイスへの接続を行うもので、ベンダ固有情報を理解するI/Oドライバともいえます。DAサーバがアクセスできる物理デバイスのデータアイテム（DAアイテム）にはベンダが名前をつけてリストのように登録します。このアイテムの名前はフラット型でも、また、階層型（例：Tank1.Pump.PV）にすることもできます。また、DAアイテムの名前にはサーバが実際にデータを検出するためのアクセスパス名と、識別するためのアイテム定義をベンダが指定することができます。そのほか、後述のDAグループの管理や2種類のDAインタフェースの提供などを行っています。DAサーバは複数のDAクライアントと接続することができます。

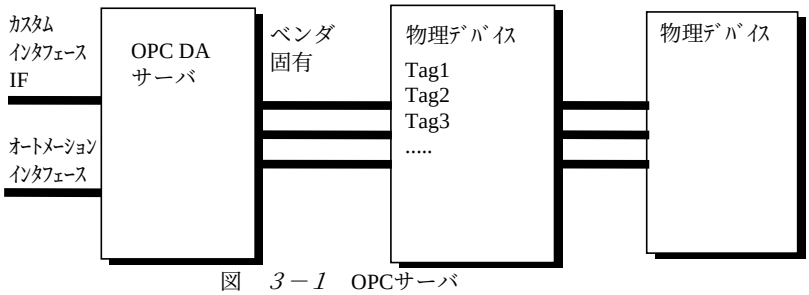


図 *3－1* OPCサーバ

DAアイテムは上述のようにサーバが認識できるように定義され、通常タグに相当するような単一の変数（セットポイントやプロセスデータ）のデータ供給元にリンクするものです。

DAクライアントはDAアイテムのレベルでメソッドをアクセスすることによりデータの読み込み／書き込みを1点ずつ行うこともできますが（VBの場合）、さらに効率よく、また、目的別にデータをアクセスできるようDAサーバ内にDAグループを1つ以上設けることができます。DAクライアントはDAグループ内に任意の個数のDAアイテムを追加することができ、図 3-2 に示すようDAグループに対して処理を要求することで一括してデータ（例：Tag1－3）を読み込み／書き込みすることができます。

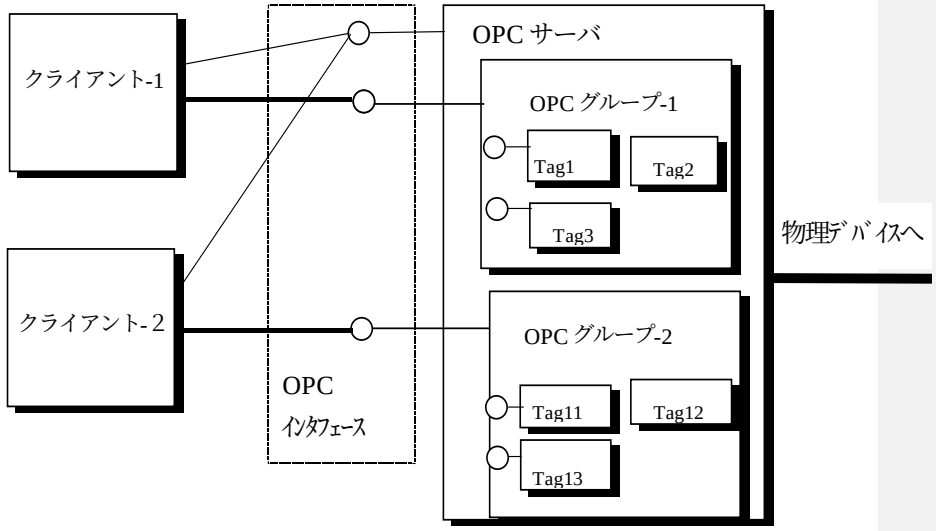


図 3－2 DAコンポーネント構成の例

3．2．1 アイテムID

アイテムを識別する任意サイズのUNICODE文字列です。いわゆるタグと呼ばれるものです。シンタックスはベンダ依存です。

DCSの例：TIC101.PV

PLCの例：COM1.STATION:42.REG:40001;0,4095,-100.0,+1234.0

ただし、DAではアイテムIDについてフラットモデルと階層構造モデルが規定されており、次のような名前の付け方ができるようになっています。（フラット/階層構造いずれをサポートするかはサーバ依存）

フラット ：TIC101

階層構造：AREA1.REACTOR3.TIC101

3．2．2 プロセスデータ

プロセスデータは表3-2に示す3種類のデータで構成されます。通常、この3つのデータは組みで扱われます。

表3-2 プロセスデータ

データ	説明	備考
データ値(Value)	値そのもの	VARIANT型
品質フラグ(Quality Flag)	データの有効性などを表わす	FieldBus Foundation仕様と同様
タイムスタンプ	データの取得時刻	UTC表現

3．2．2．1 データ値（Value）

DAにおいてはあらゆるデータ型を安全に扱えるようにプロセスデータはVARIANT型と呼ばれるデータ型を使用しています。

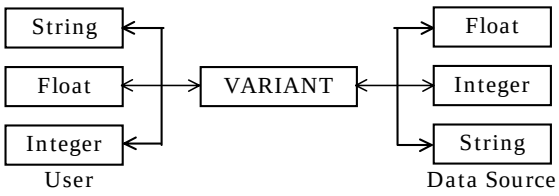


図 3-3 DAのデータ型(例)

表3-3の範囲のデータ型を自由に扱えます。

DAサーバの返す値のデータタイプはDAクライアントから要求することができます。DAサーバが、要求されたデータタイプをサポートしていない場合は警告が返され、返される値はDAサーバがそのアイテムに対して標準で規定するデータタイプ(正規化データタイプ)となります。

表 3-3 VARIANT型データ(一部)

データ型	バイト数	説明
VT_I2	2	単精度(16ﾋﾞｯﾄ)整数
VT_I4	4	倍精度(32ﾋﾞｯﾄ)整数
VT_R4	4	単精度(32ﾋﾞｯﾄ)浮動小数
VT_R8	8	倍精度(64ﾋﾞｯﾄ)浮動小数
VT_CY	8	VT_UI8と同じ 通貨(10000倍)
VT_DATE	8	VT_R8と同じ 1899/12/30からの通算日時
VT_BSTR	可変	文字列 VT_UI4で示された文字数の後にUNICODE文字とNULLターミネータを付加
VT_BOOL	1	VT_UI1と同じ 0:FALSE、－1:TRUE
VT_UI1	1	符号なし文字
VT_ARRAY	可変	上記データタイプの一次元配列

3．2．2．2品質フラグ(Quality Flag)

データ値(Value)が正しく取得できたかどうかを示すフラグです。異常時にはその原因を通知します。たとえば、デバイスやセンサの故障を検出した、あるいはアイテムが削除されている等のステータスが通知されます。この詳細についてはDA仕様書7.8を参照ください。

3．2．2．3タイムスタンプ (Time Stamp)

データが取得されたUTC時刻を表すFILETIME構造体(8バイト)データです。デバイス上の時刻、またはDAサーバ上の時刻を表します。このデータ型で表現できる時刻精度は100nsです。なお、非同期読み込みの場合はこの項目の取得を省略できます。

3．2．3アイテムの属性

アイテムの属性を表に示します。これらの属性の取得/設定はDAクライアントから任意の時点で行えます。ただし、工業単位についてはDAサーバが提供する機能であり、取得のみサポートされています。

表 3-4アイテムの属性

属性	値	説明
Active	TRUE/FALSE	FALSEにするとデータ読み込みは行われない。
アクセス権	読み込み可能/書き込み可能	そのアイテムが読み込み/書き込みできるかどうかを示す。セキュリティとは別個の、そのアイテム本来の性質。
要求データ型	データ型	DA仕様では読み込みデータのタイプは読み込み要求時にDAクライアントが指定できる。
正規化データ型	データ型	DAサーバに保持するデータのタイプ。
工業単位情報タイプ	なし、アナログ、列挙	次項目の工業単位情報のタイプを示す。(オプション)
工業単位情報	アナログ：レンジLOW/HI指定 列挙：値の0,1,2..に対応して文字列を指定(例：OPEN, CLOSE, IN TRANSIT,...)	工業単位情報タイプにより内容が異なる。(オプション) 列挙の場合、DAサーバがローカライゼーションをサポート可能。
アクセスパス	文字列(サーバ依存)	プロセスデータアクセスに複数の方法がある場合、そのいずれを使用するか要求時に指定できる。(オプション)

3．2．4グループの概念

DAクライアントが効率よくプロセスデータをアクセスする仕組みとして“グループ”が用意されています。グループにはDAクライアントが任意の個数のアイテムを登録することができ、プロセスデータアクセスは通常、これを単位として行います。グループにはActive属性や更新周期の属性などがあります。

3．2．5データのキャッシュ

同期型読み込み、非同期型読み込み要求では（要求時Call単位で）データソースとしてCACHEあるいはDEVICEのいずれかを指定します。

パフォーマンスの問題から通常はCACHEを使用します。DEVICEは診断などの限られた場合での使用となります。DEVICEのときはグループおよびアイテムのActive属性(アクティブ／非アクティブ)に関わらずデバイスからの読み込みを行います。

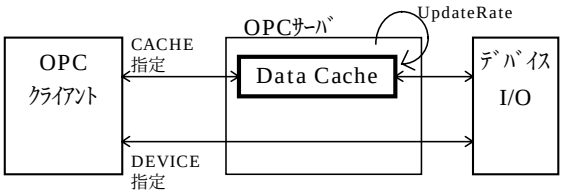


図 3-4 CACHEとDEVICE

キャッシュの更新には次の2つの場合があります。

DAクライアントからのリフレッシュ要求ではグループ内のすべてのアクティブなアイテムについてキャッシュ値を更新します。

また、DAクライアントからの要求とは無関係に一定の更新周期(UpdateRate)でキャッシュが更新されます。

UpdateRateはグループ毎に設定される属性で、DAクライアントからの要求により、指定された更新周期にできるだけ近い値がDAサーバで設定され、実際に設定された更新周期はDAクライアントに通知されます。

なお、書き込み要求に対してはDAサーバは実際のデバイスへの書き込み完了（または失敗確認）まで実行するためキャッシュ動作は無関係です。

3．2．6データ変化通知(subscription)

表2-1におけるプロセスデータ読み込みの4方式のうち、同期読み込み／非同期読み込み／リフレッシュはDAクライアントからの要求によるものですが、DAクライアントの要求によらずDAサーバから通知を受ける方式がsubscriptionです。更新周期(UpdateRate)によるキャッシュ値更新の際、値に変化があればDAクライアントに通知されます。ただし、DAサーバがデッドﾊﾞﾝﾄﾞ(PercentDeadBand)をサポートしていて、アイテムの工業単位情報タイプがアナログの場合は、現在値と前回値の差の絶対値がこれで決まる値を超えた時のみキャッシュデータを更新し、DAクライアント側のメソッドに通知されます。これによりアナログ値の小さなじょう乱を無視することができ、サーバおよびクライアントの負荷軽減となります。

3．3 OPC Data Accessのオブジェクトモデル

OPC Data Access 仕様の特長のひとつとして、DAサーバ、DAグループ、DAアイテムのオブジェクトモデルを採用していることが挙げられます。

DAサーバオブジェクトはDAサーバ内の詳細をDAクライアントから隠蔽化します。これにより、DAサーバを利用するアプリケーション(DAクライアント)は異なるベンダから提供されるDAサーバを同じ手順で使用することが可能となります。DAアイテムオブジェクトは制御する物理デバイスなどのハードウェア情報を隠蔽化します。これによりベンダ毎の物理デバイスの仕様の違いに対応できます。
DAグループオブジェクトはこれらのオブジェクトの中で最も優れたメリットがあります。上記の2つのオブジェクトとは性格が異なり、DAサーバ側で各DAクライアントの情報を保持しています。たとえば、データの変更通知レートであるUpdateRate値などを各DAクライアント毎に保持しサーバ側で管理できるので、クライアント／サーバ間での通信負荷を減らすことができ、パフォーマンスの向上を計れます。

3．3．1 DAサーバオブジェクト

DAサーバが最初に生成するオブジェクトです。DAサーバオブジェクトは、DAサーバ全体(共通)の管理情報およびDAクライアントとのコネクションにおける情報を保持しています。 DAグループオブジェクト、DAアイテムオブジェクトは、このDAサーバオブジェクトと関連してつくられます。

項目	説明	備考
スタート時刻	DAサーバ起動時刻	UTC
現在時刻	DAサーバの現在時刻	UTC
バンド幅	使用率など	サーバ依存
メジャーバージョン	DA サーバソフトウェアのメジャーバージョン番号	
マイナーバージョン	DAサーバソフトウェアのマイナーバージョン番号	
ビルド番号	DA サーバソフトウェアのビルド番号	
ベンダ情報	サーバベンダの文字列情報	ベンダ名とサポートするデバイスなど
ステータス	DAサーバの現在の状態	正常、エラー、構成情報なし、テストモードなど(DA仕様書7.7.5参照)
グループ数	DAサーバが管理しているグループ数(プライベートも含む)	主として診断用
前回更新時刻	このDAクライアントに対して値を更新した前回時刻	UTC

表 3-5 DAサーバの属性

1つのDAサーバオブジェクト内には1つ以上のDAグループを定義できます。1つのグループの中には、1つ以上のDAアイテムを定義できます。図に示すと以下のようになります。

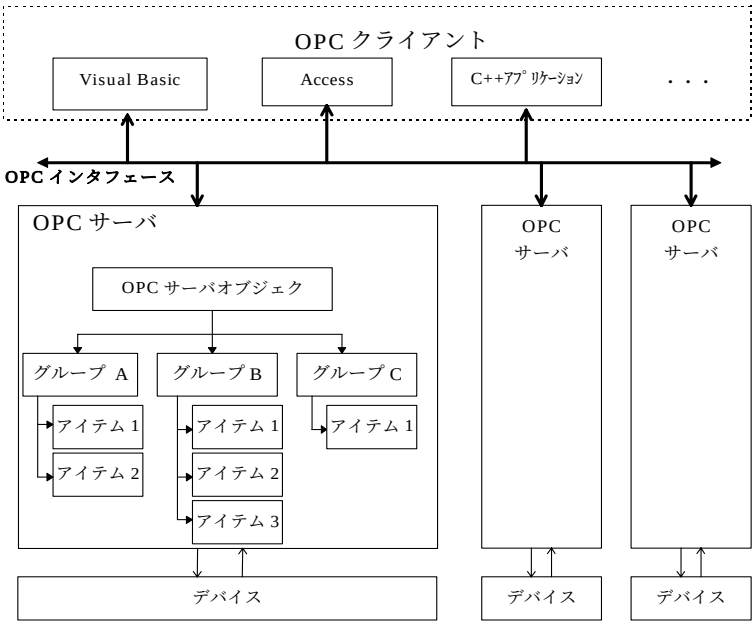


図 3-5 DAサーバオブジェクトの内部

3．3．2 DAグループオブジェクト

DA グループオブジェクトがアイテムの集合を管理するために提供するオブジェクトです。データの入出力は基本的に DA グループ単位で行います。このため、DA クライアントは DA グループ単位でアクティブか非アクティブの設定など、グループ単位でのデータ入出力の制御を行うことができます。表 2-6 に DA グループの属性を示します。DA クライアントはこれらの属性の取得／設定ができます。

表 3-6 DAグループの属性

属性	説明	備考
グループ名	この DA クライアントグループの中でユニークな名前	パブリックグループ(後述)の場合はすべてのパブリックグループの中でユニークな名前。
Active	アクティブ／非アクティブ	
UpdateRate	データ変更通知のレート	単位はミリ秒。
Time Zone(Time Bias)	デバイスの時刻との差分調整用	データのタイムスタンプをデバイスの Local time に戻す時に使用される。単位は分。
デッドバンド	不感帯（百分率）	工業単位情報タイプがアナログの時
LCID	DAサーバが値を文字列として返す時に用いられる言語識別子	工業単位列挙を含むアラームやステータスなど

3．3．3 プライベートグループとパブリックグループ

通常、グループはプライベートグループとして作成されます。プライベートグループはそれを作成したDAクライアントのみがアクセスできますが、複数のDAクライアントからアクセスできるパブリックグループもオプションとして用意されています。この場合、最初、プライベートグループとして作成されたグループをMoveToPublicメソッド(5.3節および5.4節参照)によってパブリックグループに変更します。[1]

DAクライアントは、一度パブリックグループとして接続すると、プライベートグループと同じように簡単に操作することができます。グループやグループ内のアイテムを実行/停止させたり、クライアントハンドルをセットしたり、アイテムに要求データタイプを設定することができます。これらのすべての操作はそのDAクライアントからのみ影響を及ぼし、そのパブリックグループへ接続されたほかのDAクライアントの動作には影響しません。例外はアイテムの追加および削除ができないことです。

クライアントごとに保持される属性

グループ属性のうち、Active属性、UpdateRate、TimeZone
アイテム属性のうち、Active属性、要求データタイプ

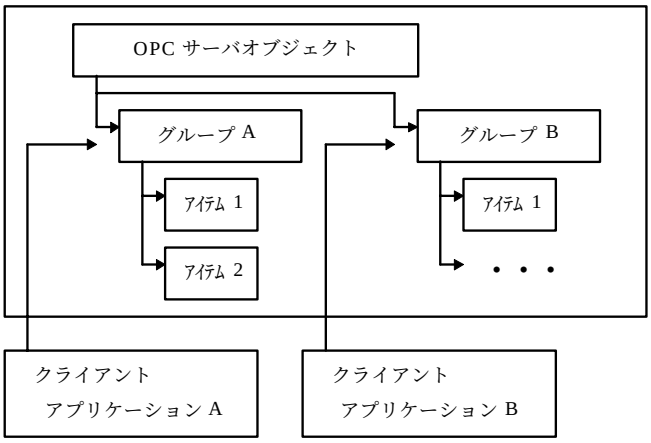


図 3-6 通常のDAグループのアクセス例

[1] OPC仕様では、クライアントからの要求によらずOPCサーバも作成できる。

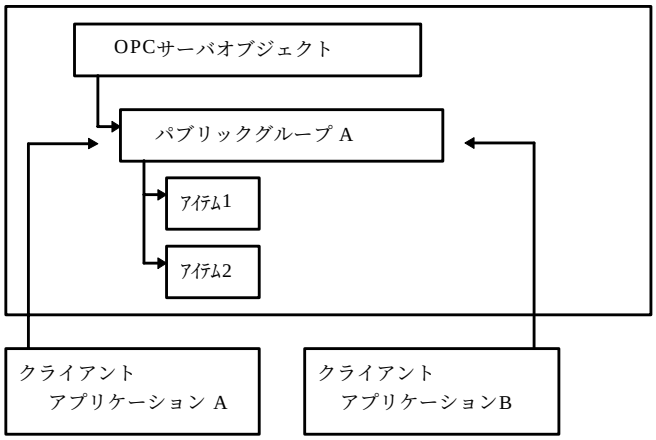


図 3-7 DAパブリックグループのアクセス例

3．3．4 DAアイテムオブジェクト

DA オートメーションインタフェースではアイテム単位でのアクセスが可能なため、DA オートメーションインタフェースに限り DA アイテムオブジェクトがあります。

3．4 ブラウズ機能

ブラウズ機能とはその内容を閲覧する機能です。これによってDAクライアントはたとえば、そのDAサーバでどのようなアイテムが利用可能か、DAサーバに現在どのようなグループがあるのか、あるいはそのグループにどのようなアイテムが登録されているのか、などを知ることができます。

3．4．1 サーバアドレススペースのブラウズ（オプション）

そのDAサーバで利用可能なアイテムをブラウズすることができます。このとき、アイテムID、データ型、アクセス権それぞれにフィルタをかけられます（このフィルタ仕様はベンダ依存）。階層構造モデルの場合はさらにブラウズ位置の変更ができ、その時のブラウズ位置から上/下方向でアイテムをブラウズできます。

3．4．2 グループのブラウズ

グループに関してはブラウズのために下記の列挙(Enumeration)機能が規定されています。

そのDAクライアントが接続しているプライベートグループの列挙
そのDAクライアントが接続しているパブリックグループの列挙
そのDAクライアントが接続している全グループの列挙
そのDAクライアントが作成したすべてのプライベートグループの列挙
DAサーバで利用可能なすべてのパブリックグループの列挙
すべてのプライベートグループとすべてのパブリックグループの列挙

3．4．3 アイテムのブラウズ

アイテムに関してはブラウズのために下記の列挙(Enumeration)機能が規定されています。

グループ中のアイテムのブラウズ
指定アイテムのアクセスパスのブラウズ

3．5　構成情報のセーブ･ロード機能

3．5．1 サーバアドレススペースと構成情報

サーバアドレススペースの定義方法についてはDAでは規定していませんが、例として次のような方法を挙げています。
･完全に固定
･DA環境の外で完全に決める
･”intelligent”なDAサーバの立ち上げ時に自動的に定義（ポーリングして）
･DAクライアントアプリケーションが要求しているアイテムの名前に基づき、”intelligent”なDAサーバがその場で定義する

これとは別にDAサーバ全体の構成情報については、それらの情報のセーブ/ロードのインタフェースが用意されており、これを利用してDAサーバ立ち上げ/シャットダウン時での構成情報の保持ができる仕組みが作られています。これは、グループやアイテムの定義などクライアント特有の情報を対象にしてはいません。

3．5．2 構成情報のセーブとロード（オプション）

DAサーバが保持する構成情報はIPersistFileインタフェースを実装することによりファイルセーブ／ロードができます。（IPersistFileインタフェースはOLE標準インタフェースのひとつで、これを実装するかどうかはベンダに依ります）

3．6　その他

3．6．1 文字コード

DA インタフェースでの文字列パラメータはすべて UNICODE と規定されています。(Windows NT の文字コード体系をそのまま利用している)。

3．6．2 NLS(National Language Support)

DA クライアントはグループの属性として LCID を設定できます。DA サーバが文字列データを返すとき、この LCID を参照して処理を行うことができます。(サーバ依存)

4 OPC構造と動作

本章ではOPC仕様に従ったサーバおよびクライアントの構造と動作について説明します。

〔注意〕OPCサーバの実行環境はWindows NT 4.0 以上が必要となります。また、OPCクライアントの実行環境はWindows NT 4.0以上またはWindows 95 (Remote ServerへのアクセスにはDCOM95が必用)・98が必要となります。以下の説明はWindows NT 4.0の使用を前提にします。

4.1 OPCサーバの構造

OPCサーバの構造にはIn-Proc Server (DLLサーバ)、Local / Remote Server (EXEサーバ)の実装形態があり、さらにパフォーマンス向上のためにLocal / Remote ServerにHandlerを併用した構造とすることが考えられます。

4.1.1 In-Proc Server

In-Proc ServerとはOPCサーバを利用するクライアントアプリケーションのプロセス空間内で動作するサーバです。すなわち、DLLとして実装されたサーバで、OPCクライアントからのOPCサーバオブジェクト生成要求があると、自動的にOPCクライアントプロセス内にサーバプログラムがロードされて実行されます。In-Proc Serverの構造を図 4--1に示します。

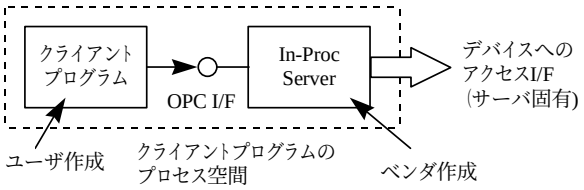


図 4--1 In-Proc Serverの構造

4.1.2 Local / Remote Server

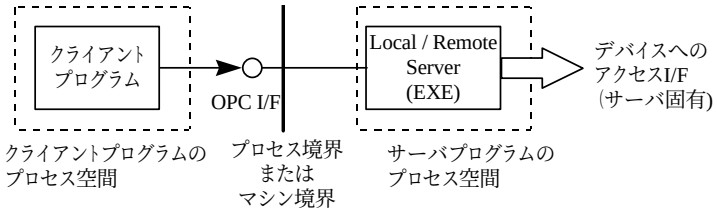
Local / Remote ServerとはOPCサーバを利用するOPCクライアントアプリケーションとは別なプロセスとして動作するサーバです。すなわち、EXEプログラムとして実装されたサーバで、OPCクライアントからのOPCサーバオブジェクト生成要求があると、自動的に該当プログラムが実行され、サーバプロセスが生成されます。Local / Remote Serverの構造の概略を図 3-2 (a)に示します。

Local / Remote Serverでは複数クライアントから接続要求があった場合、新たなサーバプロセスを生成せず単一のプロセス内で全てのOPCサーバオブジェクトを作成することが可能です。この構成ではサーバ内で物理デバイスに対する複数アクセスのロックを実現できます。

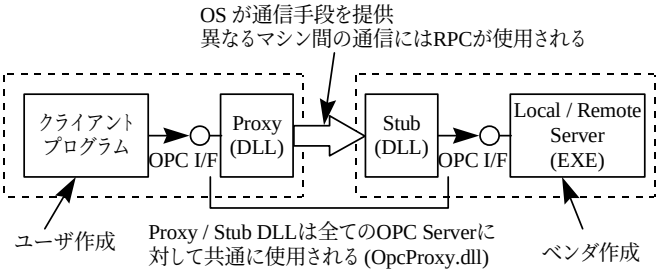
サーバの動作するマシンはクライアントと同じマシン(Local Server)でも異なるマシン(Remote Server)でも可能です。異なるマシン間の接続にはDCOM (分散COM)によるRPC通信が使用されます。サーバから見た場合、同一マシン内での接続もリモートマシンからの接続も全く同様に行われるため、サーバプログラム内で接続形態を意識する必要はありません。

Local / Remote Serverとクライアントは異なるプロセス空間で動いているため直接、メソッドの呼び出しやデータ交換を行うことはできません。実際には、図 4-2 (b) に示すように各々のプロセスにProxy / Stubと呼ばれるDLLがロードされ、OSの提供する通信機能を用いてメソッドの呼び出しやデータ交換を行います。

OPCカスタムインタフェースの場合、Proxy / Stub DLLは全てのサーバに共通に使用されます。このDLLはOPC仕様書付録のOPC.IDLから、MIDLと呼ばれるツールにより生成されます(OpcProxy.dll)。また、OPCオートメーションインタフェースの場合、OS標準のProxy / Stub DLL (oleaut32.dll)が使用されます。



(a) Local / Remote Serverの概念図



(b) Local / Remote Serverの実際の構造

図 4-2 Local/Remote Serverの構造

4.1.3 Handlerの併用

Local / Remote Server (EXEサーバ)の場合、プロセス間通信が行われるため、In-Proc Server (DLLサーバ)に比べると低速です。

そこでパフォーマンス改善のため、Handlerと呼ばれる一種のサーバを併用することがOPC仕様では推奨されています。Handlerはクライアントと Local / Remote Serverの中間に位置し、サーバからのデータをキャッシュするなどの機能を持ちます。Handlerはクライアントプロセス空間内で動作し、Local / Remote Serverへのプロセスの切り替えやプロセス間通信、ネットワーク通信によるオーバヘッドを軽減することができます。

一般的に、Handlerは各Local / Remote Serverに対応してデータアクセスを最適化したものをベンダが作成して提供します。構造としてはIn-Proc Serverと同様で、Handlerの中から実際のLocal / Remote Serverに接続します。

なお、一般的なOLEの Handlerはクライアントからの特定のEXEサーバに対するアクセス時にシステムによって自動的にロードされます。(レジストリの該当サーバのInProcHandler32エントリに登録される。) しかし、OPC仕様書で規定しているHandlerは通常のDLLサーバと等価で、クライアントはHandlerを明示的に接続する必要があります。

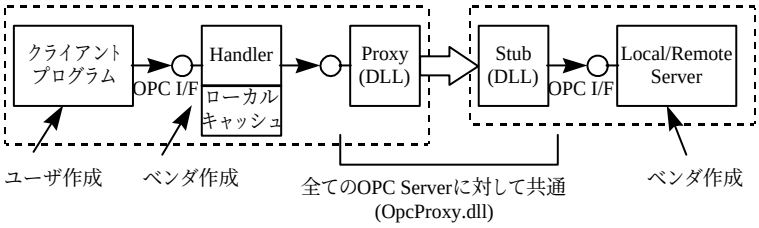


図 4-3 Handlerを併用した構造

サーバの構造によるパフォーマンスの違いについては次の記事を参照してください。この記事のデータによれば、Local / Remote Serverに対して、Handlerを併用することによりかなりの性能改善が期待できることがわかります。

Microsoft System Journal 1995 December “OLE Q&A” (日本語版 1996年2月号)

4．2 OPCカスタムインタフェースとOPCオートメーションインタフェース

OPC仕様はOPCカスタムインタフェースとOPCオートメーションインタフェースと呼ばれる2種類のインタフェースを規定しています。データアクセスの機能としては2つのインタフェースはほぼ同等の機能を備えていますが、想定しているクライアントプログラムが異なります。

OPCサーバは、通常、両方のインタフェースを実装し、いずれの種類のクライアントプログラムにも対応できるようにします。

2つのインタフェースの比較を表3-1に示します。

表 4-1OPCが規定するカスタムインタフェースとオートメーションインタフェースの特長

	カスタムインタフェース	オートメーションインタフェース
用途	SCADA / MES / 解析ソフトなどの専用アプリケーション向け	スクリプト言語からの簡便なアクセス向け
クライアントの使用言語	C / C++	Visual Basic, Delphiなど (C / C++からのアクセスも可能)
パフォーマンス	○	△
単一アイテムデータアクセス	△	○

OPCカスタムインタフェースはOLE / COMの基本的な機能をそのまま用いたインタフェースで、高速に動作します。

OPCオートメーションインタフェースはVisual Basic等からのアクセスを実現するOLEオートメーションインタフェースに従っています。スクリプト言語からのアクセスを容易にするための処理がオーバヘッドとなり、OPCカスタムインタフェースよりもパフォーマンスが劣ります。

オートメーションインタフェースの実装方法は複数ありますが、OPC仕様では"Dual"と呼ばれる実装方法を使用することが決められています。Dualの実装方法に従ったサーバは、Visual BasicなどのDual対応のオートメーションコントローラから呼び出された場合、IDispatch (通常のOLEオートメーション呼び出しで使用されるインタフェース)呼び出しではなく、VTABLE参照と呼ばれる呼び出し方法が使われます。これにより、OPCカスタムインタフェースに近い、高いパフォーマンスが得られます。

2つのインタフェースのパフォーマンスの違い、およびDual実装による高速化の効果は、前節で参照したMicrosoft System Journalの記事を参照してください。

4．3 OPCクライアントの構造

OPCクライアントは複数のベンダから提供されたOPCサーバを任意の個数アクセスすることが可能です。

OLE / COMの仕様により、サーバがIn-Proc Serverなのか、Local Serverなのか、Remote Serverなのかを、OPCクライアントでは全く意識する必要がありません。

また、Handlerに対しては通常のIn-Proc Serverと同様にクライアントからアクセスします。実際のデバイスに対する処理を行うLocal / Remote Serverへの接続はHandlerが行います。

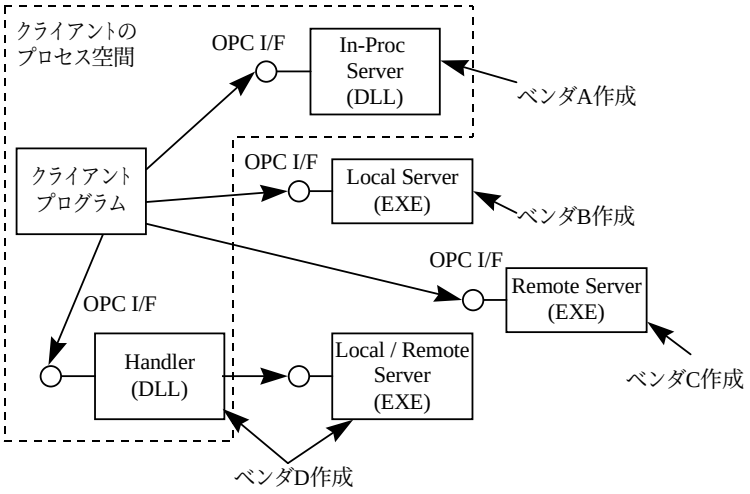


図 4-4 OPCクライアントの構造

OPCサーバはそのインストール時にサーバに関するレジストリエントリ内にOPCという名前のキーを作成することになっています。OPCクライアントはレジストリからこのキーを検索することで利用可能なOPCサーバをリストアップすることができます。

4.4 データの読み込み（Data Access 仕様）

4.4.1データのキャッシュ

DA仕様ではデバイスの実データを一旦DAサーバ内のバッファに格納する構造を想定しています。図 4-5にこの構造を示します。DAサーバはグループ単位で設定された更新周期(UpdateRate)ごとに実際のデバイスからデータを取得しバッファに保存します。

更新周期はグループごとに設定される値です。DAクライアントはDAサーバに対して更新周期の希望設定値を申告します。DAサーバは希望値にできるだけ近い更新周期を実際に設定します。実際に設定された更新周期をDAクライアントに対して通知します。

グループやアイテムはActive属性を持ち、DAクライアントが設定できます。この属性がFALSEに設定されるとそのグループ/アイテムのキャッシュデータ更新を行いません。各時点でDAクライアントが使用しないグループ/アイテムに対してFALSEを設定することにより、DAサーバの負荷を軽減することができます。

オプションの仕様である”Percent Deadband”パラメータをDAサーバがサポートする場合、DAサーバは更新周期ごとにキャッシュ保存値とその時点での実データを調べ、変化幅の割合が指定した値よりも小さかったときにはキャッシュデータを更新しないという動作になります。

DAインタフェースの読み込みメソッドのパラメータには”CACHE”と”DEVICE”という属性があります。”CACHE”が指定されたとき、DAサーバはバッファにあるデータをDAクライアントに

返します。このため処理速度は高速です。一方、”DEVICE”が指定された場合は、その都度デバイスに直接アクセスしてデータを取得します。

なお、以上のキャッシュの構造と動作は“強く推奨”されているものですが、必ずしも守らなければならない仕様ではありません。

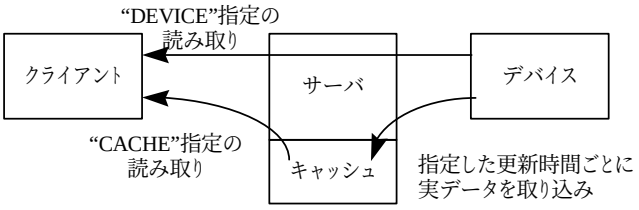


図 4--5 DAサーバの読み込みとキャッシュ

4．4．2 読み込みデータの値

DAサーバからDAクライアントに返されるデータ値 (Value)はVARIANT型の変数で扱われます。VARIANT型とはデータ実体とそのデータタイプを示すフラグを収めた自己記述型のデータ型です。

DAサーバの返す値のデータタイプはDAクライアントから要求することができます。DAサーバが要求されたデータタイプをサポートしていない場合は警告が返され、返される値はDAサーバがそのアイテムに対して標準で規定するデータタイプ(正規化データ型)となります。

4．4．3 読み込み機能

DAインタフェースではデータの読み込みについて同期読み込み、非同期読み込み、リフレッシュ、Subscriptionの4種類の方法を定めています。DAクライアントはこれら4種類の読み込みをどのように組み合わせて使用してもかまいません。DAサーバ側ではこの4種類の呼び出しが相互に影響を与えないように設計する必要があります。

同期読み込みを除く読み込み方法ではデータ転送のためにDAサーバとDAクライアント間に通信路が必要となります。これにはOLE / COMが標準で規定しているIConnectionPointContainer / IOPCDataCallbackインタフェースによる接続が使用されます。DAクライアント側はIOPCDataCallbackインタフェースを実装し、DAサーバのIconnectionPointContainer:: EnumConnectionPoints またはIconnectionPointContainer:: FindConnectionPointメソッドでこれを登録し、あらかじめ接続を確立しておく必要があります。

IOPCDataCallback経由のデータ転送路は各読み込み要求の応答で共通に使用されます。このため、転送されるデータの先頭にはトランザクション IDが付され、どの呼び出し要求に対応したデータなのかをDAクライアント側で判断することができるようになっています。

コメント [‹T“c1]:

4．4．3．1 同期読み込み (IOPCSyncIO::Read)[2]

DAサーバはDAクライアントからの要求に対してメソッドの引数にデータを格納して返します。DAクライアントはそのメソッドがリターンするまで待ち状態となります。

DAクライアントは読み込み方法としてCACHEまたはDEVICEを指定します。CACHE指定の場合、グループおよび各アイテムのActive属性がTRUEになっているものに対してのみ有効なデータを返します。FALSEの場合は品質フラグが“データ無効”を示します。DEVICE指定の場合、Active属性とは関係なく実データを返します。

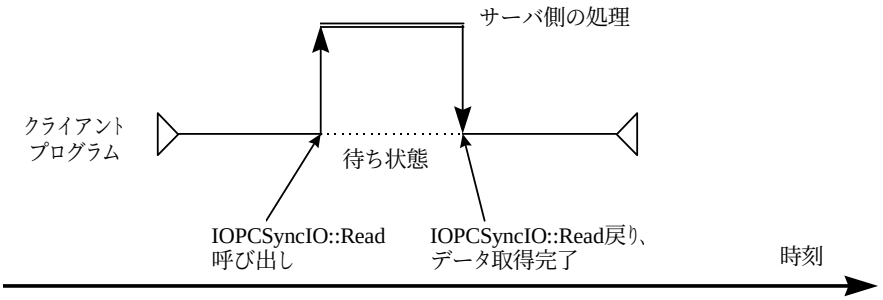


図 4-6. 同期読み込みのタイムチャート

4．4．3．2 非同期読み込み (IOPCAsyncIO2::Read)

DAサーバは要求を受付けた後すぐにメソッドをリターンします。DAクライアントはこの時点で他の処理を継続して実行できます。DAサーバ側は要求された全アイテムのデータを返す準備ができた時点でDAクライアント側のIOPCDataCallback::OnReadComplete メソッドを呼び出し、データを転送します。

DAクライアントは読み込み方法としてCACHEまたはDEVICEを指定します。CACHE指定の場合、グループおよび各アイテムのActive属性がTRUEになっているものに対してのみ有効なデータを返します。FALSEの場合は品質フラグが“データ無効”を示します。DEVICE指定の場合、Active属性とは関係なく実データを返します。

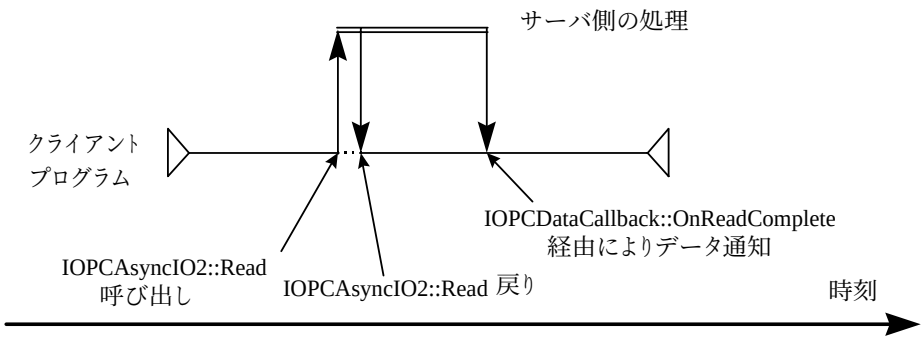


図 4 -7. 非同期読み込みのタイムチャート

[2] IOPCSyncIOインタフェースのReadメソッドを意味します。以下同様。

4．4．3．3 リフレッシュ (IOPCAsynIO2::Refresh)

DAサーバはこのメソッドが呼び出された場合、グループ内のすべてのActiveなアイテムについてそのキャッシュ値を更新し、変化のあったアイテムのデータをDAクライアント側のIOPCDataCallback::OnDataChange メソッドを呼び出して転送します。

インタフェース規定上、読み込み方法としてCACHEまたはDEVICEが指定できますが、意味があるのはDEVICE指定の場合のみです。DEVICE指定でこのメソッドが呼び出されると、グループのActive属性には関係なく、グループ内のActiveなアイテムについてデバイスからデータを取得してキャッシュ値を更新し、データ転送を行います。

本メソッドの実行はグループのActive属性に無関係なため、グループをInactiveとした上でデータが必要なときに本メソッドを呼び出すことが可能です。このようにすると、メソッド呼び出し時にのみデータが更新･転送されるのでサーバの負荷が軽減されます。次節で説明するSubscription動作は不要だが非同期の読み込み動作が必要というクライアントの要求に対してはこの使用方法が適しています。

本メソッドによるキャッシュ値更新はグループの更新周期(UpdateRate)に基づくキャッシュ更新の自動実行タイミングには影響を与えません。たとえば更新周期が1時間に設定されていて、自動更新から45分後にリフレッシュを実行したとすると、さらにその15分後にキャッシュの更新が自動的に実行されます。

4．4．3．4 Subscription

DAサーバはグループの更新周期(UpdateRate)ごとにDAサーバ内のキャッシュ値を更新します。DAクライアント/サーバ間にIConnectionPointContainerによる接続が確立されている場合、DAサーバはキャッシュ更新時にデータ値の変化をチェックし、変化があったアイテムについてそのデータをDAクライアントのIOPCDataCallback::OnDataChangeへ自動的に送信します。このデータ送信はトランザクション IDとして0という特別な値を設定するため、DAクライアントはリフレッシュによるデータと区別することができます。

DAサーバが“Percent Deadband”パラメータをサポートしている場合、アナログデータに関する上記のキャッシュ更新および通知を、指定した割合を超える変化が生じたアイテムのみに限定することができます。

4．4．3．5 まとめ

表 4-2読み込み動作のまとめ

読み込み方法	ソース	グループ	アイテム	サーバの動作
同期読み込み	Cache	Active	Active	指定されたアイテムについて有効なキャッシュデータを返す。
	Cache	Active	Inactive	指定されたアイテムについて“無効”品質フラグを持つデータを返す。
	Cache	Inactive	×	指定されたアイテムについて“無効”品質フラグを持つデータを返す。
	Device	×	×	指定されたアイテムの現在のデバイス値を返す。
非同期読み込み	×	×	×	指定されたアイテムの現在のデバイス値を返す。
リフレッシュ	Cache	Active	×	すべてのアクティブなアイテムについてキャッシュデータを返す。もしアクティブなアイテムが一つも無ければE_FAILを返す。
	Cache	Inactive	×	E_FAILを返す。
	Device	Active	×	キャッシュを更新し、すべてのアクティブなアイテムについてキャッシュデータを返す。もしアクティブなアイテムが一つも無ければE_FAILを返す。
	Device	Inactive	×	E_FAILを返す。
subscription	×	Active	Active	キャッシュ値が変化したアイテムすべてのデータを送る。
	×	Active	Inactive	このアイテムのデータは送らない。
	×	Inactive	×	このアイテムのデータは送らない。

×＝いずれの設定値でも同じことを示す。

同期読み込みではメソッドの戻り値としてデータを返す。

そのほかの読み込みではDAサーバがIOPCDataCallbackのOnDataChangeまたはOnReadCompleteメソッドを呼び出してデータを送る。

4．5 データの書き込み（Data Access仕様）

4．5．1 書き込みデータの値

DAクライアントは、デバイスに対して書き込みたい値をVARIANT型変数でDAサーバに渡します。

DAクライアントから渡されたデータのデータタイプが対象アイテムに対して適切でない（DAサーバがサポートしていない）場合にはエラーとなります。

4．5．2 データの書き込み機能

DA仕様ではデータの書き込みについて同期書き込み、非同期書き込みの2種類の方法を定めています。DAクライアントはこれら2種類の書き込みを組み合わせて使用してもかまい

ません。DAサーバ側ではこの2種類の呼び出しが相互に影響を与えないように設計する必要があります。

なお、書き込み動作に関してはグループやアイテムのActive属性の値は影響を与えません。また、書き込み要求に対してはDAサーバは実際のデバイスへの書き込み完了（または失敗確認）まで実行する仕様になっており、キャッシュ動作は無関係です。

4．5．2.1 同期書き込み (IOPCSyncIO::Write)

DAクライアントは書き込みたいアイテム（複数可）の値を引数にこのメソッドを呼び出します。DAサーバは実際にデバイスへの書き込みを実行し、全ての要求された書き込みが完了（または実行失敗を確認）するまでリターンしません。DAクライアントはメソッドが完了するまで待ち状態となります。

4．5．2.2 非同期書き込み (IOPCAsyncIO2::Write)

DAクライアントは書き込みたいアイテム（複数可）の値を引数にこのメソッドを呼び出します。DAサーバは要求を受付けた後すぐにリターンします。DAクライアントはこの時点で他の処理を継続して実行できます。

DAサーバは実際にデバイスへの書き込みを実行し、全ての要求された書き込みが完了（または実行失敗を確認）した時点でDAクライアントに処理完了を通知します。

DAクライアントへの処理完了通知にはIOPCDataCallback::OnWriteCompleteメソッドの呼び出しが使われます。DAクライアントはDAサーバとの間に3．4節にて説明したIConnectionPointContainer / IOPCDataCallbackによる通信路をあらかじめ設定しておく必要があります。

4．6 データの同期 (Synchronization)

ここで同期と呼んでいるのは1回の処理で読み込みあるいは書き込みされる複数データの同時性のことをさしています。例として次のような問題があります。

- すべてのアプリケーションはDAサーバから返されるあるアイテムに対するデータの値、品質フラグ、タイムスタンプの同期がとれていることが必須と考えられます。
- バッチ処理ではレポート作成に必要な複数の読み込みデータの同期が取れていること、またバッチ開始前に必要な指示データが全てダウンロードされていることが必要と考えられます。

これらのデータの読み書きの同期はDAインタフェース単独では保証していません。

サーバをマルチスレッドとした場合、処理スレッドが待ち状態になる場合があり、その再開時にはデータの処理時刻がずれてしまいます。通常、処理の同期が必要なデータにはロックをかけて対応しますが、その分だけ他の処理待ちが長くなる可能性が高くなります。処理待ちが増えると、サーバとしての処理性能は低下します。

したがって、サーバの対象とする実際のデバイスでのデータ同期の必要性を性能とのトレードオフで十分検討し、サーバを設計することが必要となります。

4．7 データの順序保証

すべてのDAサーバは同一デバイスに対する書き込み要求に対して要求された順序に従って処理を実行します。

たとえば、バッチ処理で考えると、書き込み順序が逆転した場合、必要なデータのダウンロード前にバッチ開始フラグが立ってしまうというようなことが生じてしまいます。このため、データ書き込みの順序保証が必要となります。

DAサーバがデータの順序保証を行うことはDAのインタフェース規定として必須ではありませんが、“強く推奨”されています。

4.8 サーバのマルチスレッド化

サーバのパフォーマンスを高めるためにはマルチスレッド化するのが有効です。OPC仕様ではサーバがどのようなスレッディングモデルを用いるかは定めていませんが、その処理内容から考えてフリースレッディングモデルを適用するのが妥当です。

このモデルによるマルチスレッド化ではスレッドセーフの保証はすべてサーバの責任となります。前述のデータの同時性や書き込み順序保証などの設計に十分注意が必要です。

フリースレッディングモデルの詳しい説明は下記を参照してください。

Microsoft System Journal 1996 May “Introducing Distributed COM and the New OLE Features in Windows NT 4.0” (日本語版 1996年6月号)

5 OPC対応ソフトウェア開発

本章ではOPC仕様に準拠したOPCサーバ／OPCクライアントソフトウェア開発の概要を説明します。OPC仕様書ではあまり触れられていない、開発に必要な資格、知識、項目、環境についての概要を記述します。さらに詳しい実装方法や開発手順については、“OPC実装ガイド”に解説されています。ただし，OPC実装ガイドを入手するにはOPC-F会員になる必要性があります。

まず、OPCサーバの開発概要を説明し、OPCクライアントの開発については“5．4 OPCクライアントの開発”でまとめて説明します。

5．1 開発資格

OPCのロゴを使用したOPCサーバを製品化するにはOPC-J会員となるべきです。OPC-J会員になることにより次のような特典が得られます。

・OPCのロゴ使用と製品カタログ掲載

・サンプルコード取得

・各種ドキュメント取得

・最新情報入手

・技術サポート

なお、OPCサーバを開発するために必ずしもOPC-Jの会員となる必要はありません。公開されている仕様書に基づき開発し、販売することができます。しかし、この場合はOPCのロゴを使用して製品化することはできません。

5．2 必要な知識

OPCサーバを作成するには以下の知識を必要とします（実装に関しては前述の”OPC実装ガイド”を参照してください）。また、各項目に関しては6章の技術基盤説明を参考にしてください。

表 5-1 OPCサーバ作成に必要な知識

分 類	項 目
OLE/COMの一般知識	インタフェースの定義
	OLEコンポーネントの実装形態
	マーシャリングの概要
OLE標準インタフェースの知識	OLEオートメーションの実装形態
	コネクションポイントの実装形態
OPCの知識	OPCで規定されたインタフェースの知識

また、上記の前提とする知識以外に、OPC-Jで開催される技術セミナーを受講することを推奨します。

5．1 開発項目

3章で説明したように、OPCサーバにはC/C++用のカスタムインタフェースと，主にVisual Basic用のオートメーションインタフェースをサポートしたものが存在します。このうち，オートメーションインタフェースをサポートしたOPCサーバは，OPC-Fから提供されるautomation wrapper dllが使用可能なため，開発は不要です。ここでは，カスタムインタフェースをサポートしたOPCサーバに関する開発項目を示します。カスタムインタフェースをサポートしたOPCサーバの構造には、In-Proc Server、Local/Remote Server(In-Proc Handlerなし)、Local/Remote Server(In-Proc Handlerあり)が存在しますが，すべての形式のサーバを開発する必要はなく、サーバベンダにとって適切なタイプのサーバを用意することになります。これらの特徴については、“3章 構造と動作”を参照してください。以下にカスタムインタフェースに関して開発に必要な項目を示します。

5．1.1 In-Proc Server

(1)DLL形式のサーバ
OPC-Fから提供されるサンプルコードを利用して作成することができます。

(2)インストーラ
サーバ、タイプライブラリのインストールを行うとともに、レジストリにサーバオブジェクトの情報を登録するものです。

5．1.2 Local/Remote Server(In-Proc Handlerなし)

(1) EXE形式のサーバ
OPC-Fから提供されるサンプルコードを利用して作成することができます。

(2) Proxy/Stub DLL
OPC-Fから提供されるIDL (インタフェース定義言語) ファイルをMIDLコンパイラによりコンパイルして作成します。

(3) インストーラ
サーバ、タイプライブラリ、Proxy/Stub DLLのインストールを行うとともに、レジストリにサーバオブジェクトの情報を登録するものです。

5．1.3 Local/Remote Server(In-Proc Handlerあり)

(1) EXE形式のサーバ
OPC-Fから提供されるサンプルコードを利用して作成することができます。

(2) DLL形式のIn-Proc Handler
独自に作成する必要があります。

(3) Proxy/Stub DLL
OPC-Fから提供されるIDL (インタフェース定義言語) ファイルをMIDLコンパイラによりコンパイルして作成します。

(4) インストーラ
サーバ、In-Proc Handler、タイプライブラリ、Proxy/Stub DLLのインストールを行うとともに、レジストリにサーバオブジェクトの情報を登録するものです。

なお、サンプルコードやProxy/Stub DLL はOPC-Fより提供されております。サンプルコードは

C++で記述されたコードで基本的な機能が盛り込まれたスケルトンとなっています。
また、インストーラはOPC仕様で規定されているものではありませんが、製品としては必須のものです。これについてのサンプルコードはありません。

5．2 開発プラットホームと開発ツール

(1) 開発プラットホーム - - - Windows NT 4.0以降

OPCサーバを開発するプラットホームとしては、Windows NT4.0以降が必要です。今後、他のプラットホームでも開発できる環境が整備されるかもしれませんが、現状はWindows NT4.0のみしか動作を確認していません。なお, リモートマシンからDCOMを利用したテストを行うには, Windows NT 4.0 Servce Pack2以降が必要になります。

(2) 開発ツール - - - Microsoft Visual C++ 5.0以降

OPCサーバを開発するツールとしては、Microsoft Visual C++5.0以降が必要です。他の開発ツールでもMicrosoft Visual C++5.0と同様の機能を持つ開発ツールであれば、開発することは可能ですが、OPC-Fが提供するサンプルコードや今後、作成が計画されているOPC実装ガイドではMicrosoft Visual C++を前提に記述されることが予定されており、他の開発ツールはサポートされません。また、Microsoft Visual C++4.1以前のバージョンでは、一部、開発に支障をきたします。

5．3 認証

OPC-Fが規定する適合性テストを受けて認証を取得する仕組みが現在, 検討されています。仕組みが完成され次第, 別途アナウンスを行う予定です。

5．4 OPCクライアントの開発

製品としてOPCクライアントを開発する場合はOPCサーバを開発するのと同等の開発資格と認証が必要になります。製品としてではなく、各ベンダから購入したOPCサーバを単に利用するためのOPCクライアント開発には特に開発資格、認証の規定は適用されません。いずれの場合も必要とする知識はOPCサーバ開発の場合と比較して非常にわずかです。

OPCクライアントの開発ではOPCカスタムインタフェースで開発する場合とOPCオートメーションインタフェースで開発を行う場合で必要な知識、開発ツールが異なります。

OPCカスタムインタフェースでのOPCクライアント開発

CおよびC++言語での開発。

OPCオートメーションインタフェースでのOPCクライアント開発

Visual BasicやVBA (Visual Basic for Applications) での開発。

以下にOPCクライアントを開発する場合の必要な知識、開発プラットホームと開発ツールについて記述します。なお、各項目に関しては“6章 技術基盤説明”を参考にしてください。

5．4.1 必要な知識

表 5-2 OPCカスタムインタフェースで開発する場合の必要な知識

分 類	項 目
OLE/COMの一般知識	インタフェースの定義
OLE標準インタフェースの知識	コネクションポイントの概要
OPCの知識	OPCで規定されたインタフェースの知識

表 5-3 OPCオートメーションインタフェースで開発する場合の必要な知識

分 類	項 目
OLE標準インタフェースの知識	OLEオートメーションの概要
OPCの知識	OPCで規定されたインタフェースの知識

5．4.2 開発プラットホームと開発ツール

5．4.2.1 OPCカスタムインタフェースで開発する場合

開発プラットホーム - - - Windows 95以降あるいはWindows NT 4.0以降

OPCカスタムインタフェースでOPCクライアントを開発するプラットホームとしてはWindows 95以降あるいはWindows NT 4.0以降が必要です。今後、他のプラットホームでも開発できる環境が整備されるかもしれませんが、現状はWindows 95あるいはWindows NT 4.0のみでしか動作を確認していません。なお, リモートマシンからDCOMを利用したテストを行うには, Windows NT 4.0 Servce Pack2以降が必要になります

開発ツール - - - Microsoft Visual C++ 5.0以降

OPCカスタムインタフェースでOPCクライアントを開発するツールとしてはMicrosoft Visual C++ 5.0以降が必要です。他の開発ツールでもMicrosoft Visual C++5.0と同様の機能を持つ開発ツールであれば開発することは可能ですが、OPC-Fとして提供するサンプルコードや今後、作成が計画されているOPC実装ガイドではMicrosoft Visual C++を前提に記述されることが予定されており、他の開発ツールはサポートされません。また、Microsoft Visual C++4.1以前のバージョンでは、一部、開発に支障をきたす場合があります。

5．4.2.2 OPCオートメーションインタフェースで開発する場合

開発プラットホーム - - - Windows 95以降あるいはWindows NT 4.0以降

OPCクライアントを開発するプラットホームとしてはWindows 95以降あるいはWindows NT 4.0以降が必要です。今後、他のプラットホームでも開発できる環境が整備されるかもしれませんが、現状はWindows 95あるいはWindows NT 4.0のみでしか動作を確認していません。なお，リモートマシンからDCOMを利用したテストを行うには，Windows NT 4.0 Servce Pack2以降が必要になります

開発ツール - - - Visual Basic 5.0 Professional Edition以降およびVBA（Visual Basic for Applications）の機能を持つ開発ツール

他の開発ツールでもOLEオートメーションのクライアント機能を持つ開発ツールであれば開発することは可能ですが、OPC-Fとして提供される予定のサンプルコードやOPC仕様書の資料ではVisual Basic以外の開発ツールはサポートされません。

6 OPCアプリケーションとOPCインタフェース

6.1 OPCアプリケーション

OPCはOLE/COMの技術を基盤としたデータ交換インタフェースです。OLE/COMの技術を基盤とするOCXやActiveXコントロールへの応用などさまざまな部分で実装することが可能です。この節ではOPCアプリケーションの適用例やOPCインタフェースを使用したサンプルプログラムについて記述します。

6.1.1 アプリケーションへの適用

アプリケーションとOPCインタフェースの結合および実装についてはさまざまな方法が考えられます。プロセス制御への適用例としては、ネットワークモニタ、デバイスステータス、履歴管理、デバイス制御、ファイル、データ、プログラムのアップロードおよびダウンロードなどで実装可能です。OPC仕様に基づいたインタフェースに準拠していれば、クライアント／サーバ間のデータ交換やコンフィグレーションなどのあらゆる通信を簡単に実行することができます。次の例ではアプリケーションとハードウェアおよび外部アプリケーションとのOPCインタフェースでの適用例を示しています。

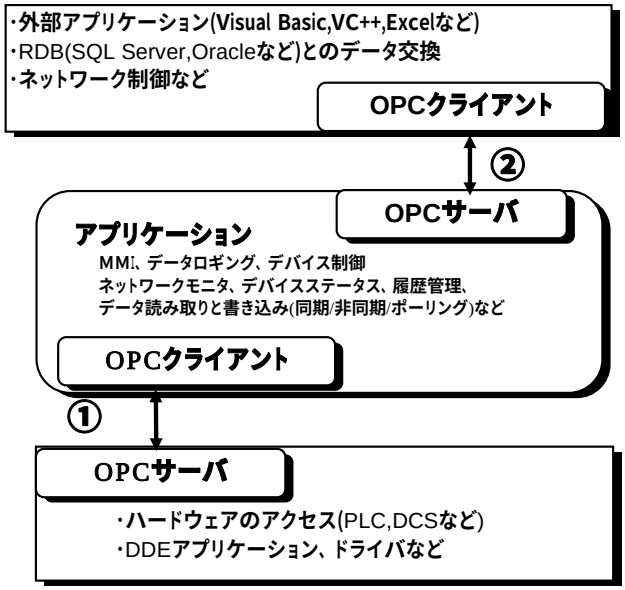


図 6-1 OPCの適用例

(1)ハードウェアとのインタフェース (①)

アプリケーションとハードウェア間のインタフェースに使用する例です。この場合、OPCサーバはハードウェアとのインタフェースを制御するI/Oドライバプログラムになります。アプリケーション（OPCクライアント）はOPCサーバへの要求により、ハードウェアのデータや設定情報などを取得、設定することができます。

(2)外部アプリケーションとのインタフェース (②)

アプリケーションと外部アプリケーションとのデータ交換を行うインタフェースの例です。この場合の外部アプリケーションとは、Visual BasicやVisual C++などで作成したカスタムアプリケーションやRDBとのデータ交換を実行する制御アプリケーションなどです。アプリケーションにOPCのインタフェース（OPCサーバ機能）を実装することにより、外部アプリケーションからのアクセスが簡単に実行できます。

6.1.2 アプリケーションからオブジェクトへのアクセス

OPCオートメーションインタフェースはOLEオートメーションインタフェースに従っており、あるアプリケーションが別のアプリケーションで実装されたコンポーネントを操作したり、別のアプリケーションが操作できるようにコンポーネントをエクスポーズ（公開）することを可能にします。OLEオートメーションオブジェクトにアクセスするアプリケーションのことを、OLEオートメーションコントローラと呼びます。OLEオートメーションコントローラは、別のアプリケーションに存在するOLEオートメーションオブジェクトを作成し、OLEオートメーションオブジェクトがサポートするプロパティとメソッドを使用してこれらのオブジェクトを簡単に操作することができます。このしくみは、オブジェクトが管理するデータ領域や機能を、あたかも自分自身のアプリケーションのものであるかのように扱えるようにしています。

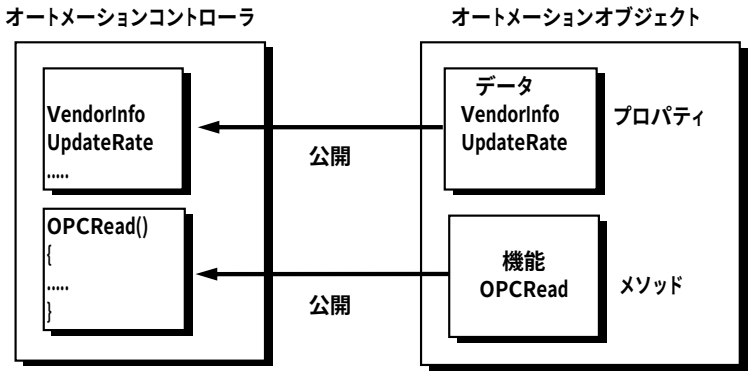


図 6-2 オートメーションコントローラとオートメーションオブジェクト

Visual Basic を使用して、オートメーションオブジェクトのプロパティに値をセットするには、

<オブジェクト変数>.<プロパティ名> = <値>

と記述します。また、オブジェクトからプロパティ値を取得するには、

<格納する変数> = <オブジェクト変数>.<プロパティ名>

と記述します。これらの構文を使用することにより、アプリケーションからオートメーションオブジェクトに簡単にアクセスすることが可能です。

Description	(Read-only) The unique identifier for this item.
Syntax	ItemID As String
Example	Dim AnOPCItem as OPCItem Set OPCItem = GetOPCItem(SomeItemServerHandle) VB Syntax Example (getting the property): Dim CurrentValue As String Dim SomeValue As String CurrentValue = AnOPCItem.ItemID

リスト 6-1 OPCオートメーションインタフェースの使用例

6.1.3 OLEコントロール、ActiveXへの応用

OLEコントロール(OCX)は、さまざまな開発環境で使用することができる、OLEを介して機能している再利用可能なソフトウェアコンポーネント（汎用性のある部品）です。OLEコントロールは機能を拡張したIn-Proc Serverです。OLEコントロールには①32ビット環境で使用できる部品規格、②データ取り込み容量の制限が大幅に緩和されている、③プロパティなどで容易にカスタマイズ可能、などの優れた特徴があります。

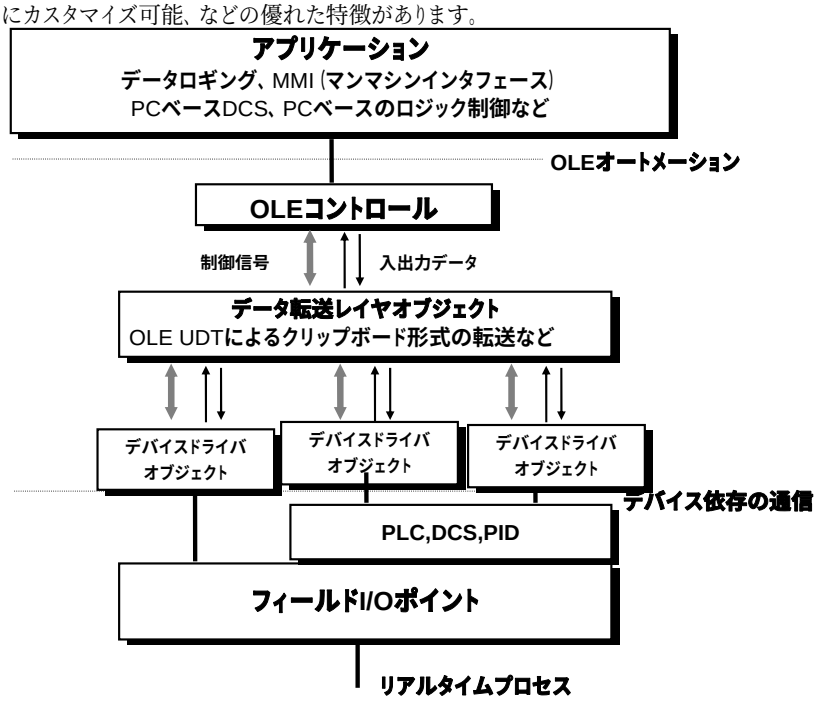


図 6-3 OPCコンポーネントを連携させたシステム構成

図6-3 はOLEコントロール、ActiveXコントロールとOLEオートメーションを使用して、OPCコンポーネントを連携させ、プロセス制御アプリケーションを構築する例です。OLEオートメーションインタフェースを使用して、各種デバイスドライバオブジェクトを経由し、各ハードウェアにアクセスしています。アプリケーションとハードウェアとのインタフェースをオブジェクトインタフェース、OLEインタフェースで標準化しています。下記に現状入手可能なActiveXコントロールの例を記載します。

ActiveXコントロール 製品例（注）	分野
ActiveBar	GUI作成支援
PerfectGrid	GUI作成支援
ChartBuilder	グラフ作成
ChartFX	グラフ作成
VS-VIEW	帳票作成･印刷
ImageGear/ActiveX	画像処理用ソフトウェア開発支援
WinRT-VB	ハードウェア制御（ポート入出力、メモリ入出力）
ComponentWorks	機器制御（データ解析･表示、計測制御）
DbComplete	データベースアクセスアプリケーション構築用
VBMan ActiveX Control for RS232C	RS232Cの通信制御コントロール
FA-Engine	機器制御

（注）「開発ツール完全ガイド'99」(日経BP社)より抜粋

このようなOLEコントロール、ActiveXコントロールとOLEオートメーションを組み合わせて開発することにより、アプリケーションで必要な機能をすべて新規に開発する必要はなくなります。すなわち、アプリケーション内で利用できるところはコンポーネントとして組み込み、それを利用できます。開発者はアプリケーション本来の機能部分の開発に専念することができます。

ActiveXはインターネットに対応するためにいろいろな工夫が追加されています。インターネット上で遠隔地のサーバからダウンロードされることを前提に設計されており、通信負荷を下げる目的で、よりモジュールサイズが小さくなるようにシンプルに設計されています。ActiveXコントロールは他のActiveXコントロールまたは任意のCOMオブジェクトで動作します。

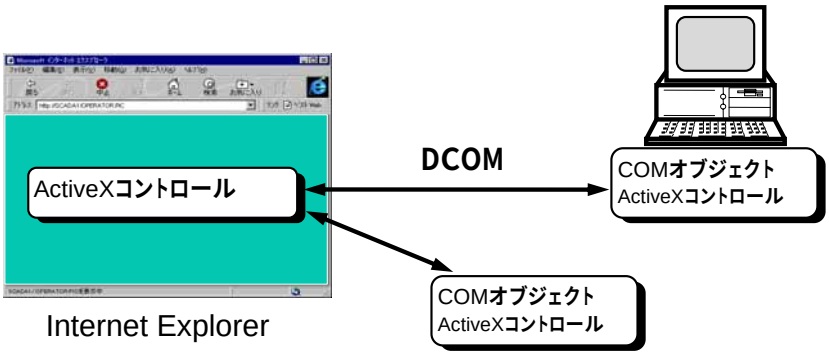


図 6-4 ActiveXコントロールの例(1)

図5-5は多数のプロセス制御用のActiveXコントロールをインターネットエクスプローラ上で表示させている例です。リアルタイムチャートやメータなどの各部品のデータ表示はネットワーク上のリモートPCからのデータを表示することも可能です。

図 6-5 ActiveXコントロールの例(2)

6．2 OPCインタフェース概要

OPCサーバオブジェクトはデータソースの特定のクラスからデータを得る方法を提供します。利用できるソースの数とタイプはサーバインプリメンテーションの関数です。

OPCサーバには“グループ“として定義される機能があります。これらのグループによって、OPCクライアントはアクセスしたいデータを取りまとめて扱うことができ、それを一つの単位として起動させたり停止させたりすることができます。さらに、グループはOPCクライアントに、変更するとき、通知できるように項目リストへ”署名”方法を提供します。

OPCインタフェースには、COMオブジェクトのインタフェースを経由してアクセスするカスタムインタフェースと、“OPCオートメーションラッパDLL”経由でCOMオブジェクトインタフェースをアクセスするオートメーションインタフェースがあります。

“OPCオートメーションラッパーDLL”は、ＯＰＣサーバのカスタムインタフェースと、オートメーションインタフェースを使用するクライアントの間を翻訳するもので、OPC Data Access Ver2.0対応として各ベンダ共通に使えるようOPC-Fから提供されるものです。

従来のＶｅｒ1．０ではオートメーションインタフェースを実現するために、ＯＰＣサーバーに実装されたIDispachインタフェースによって、カスタムインタフェースをエクスポーズしましたが、Ｖｅｒ2．０では、この“OPCオートメーションラッパーDLL”が同様の機能を提供するため、ＯＰＣサーバにIDispatchインタフェースを実装する必要はありません。

一般に、クライアントプログラムは、VisualBasic,VBAのようなスクリプト言語などを利用してオートメーションインタフェースを使用します。また、一般に、C++言語で作成される、最大のパフォーマンスを達成する目的をもつクライアントプログラムはカスタムインタフェースを使用することが最も簡単でしょう。

6．3 DAオートメーションインタフェース

OLEオートメーションは、“DA オートメーションラッパーDLL”を通して、OPCデータ アクセス カスタムインタフェースを、ほぼ忠実にエクスポーズします。実装されているカスタムインタフェースの一部のオプション機能や未実装のオプション機能を、オートメーションインタフェースがアクセスした場合には、デフォルトとして仕様書に記載されている動作をします。

DAオートメーションは、OLEオートメーションコントローラを持つアプリケーション(VB5/6、C++, VBA5/6 アプリケーション)をサポートします。ただしVBScriptやJava Scriptのサポートは範囲外です。

6．3．1 DAオートメーションオブジェクトモデル

DAオートメーションインタフェースは図 6-6のように DAサーバ (OPCServer) 、DAグループコレクションオブジェクト (OPCGroups) とDAグループ (OPCGroup) 、およびOPC DAアイテムコレクションオブジェクト (OPCItems) とそのDAアイテム (OPCItem) のオブジェクト階層から成り立っています。DAグループコレクションオブジェクト (OPCGroups) とDAブラウザオブジェクト (OPCBrowser) はDAサーバオブジェクトに含まれています。

各オブジェクトの概要を図6-6に記載します。

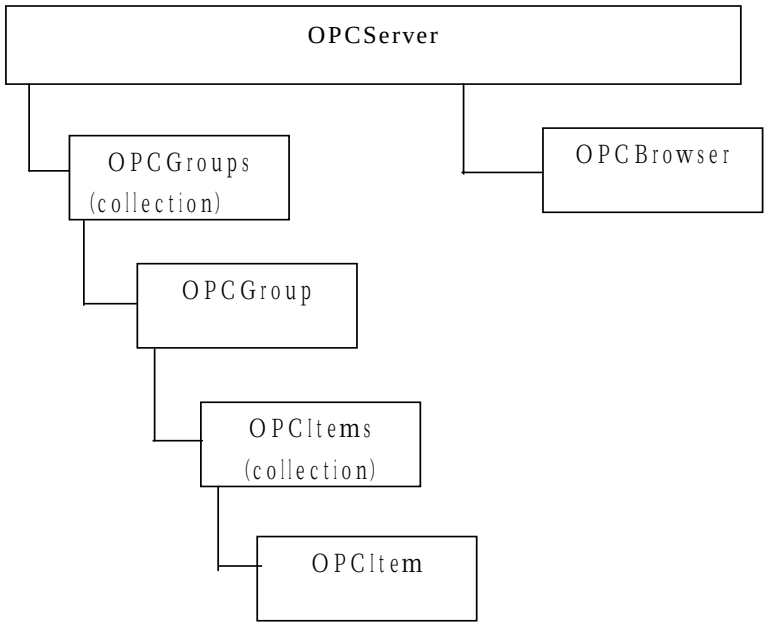


図 6-6 DAオートメーションオブジェクトの階層構造

表6-1 DAオートメーション オブジェクト モデル

オブジェクト **(名称)**	**説明**
OPCServer **(OPCサーバー)**	OPCサーバオブジェクト のインスタンスで、他のオブジェクトを参照する前に生成する必要があります。OPCGroupsコレクションを含み、OPCBrowserオブジェクトを生成します。
DABrowser **(DAブラウザ)**	DAサーバ構成中のアイテムの、名前をブラウズするオブジェクト。DAサーバーオブジェクトのインスタンスごとに1つだけ生成することができます。
OPCGroups **(OPCグループコレクション)**	クライアントがOPCサーバー中に生成したすべてのOPCグループオブジェクトのオートメーションコレクション。
OPCGroup **(OPCグループ)**	OPCグループオブジェクトのインスタンス。 ステータスの管理やOPCアイテムコレクションでのデータ収集機構を行います。
OPCItems **(DAアイテムコレクション)**	クライアントがDAサーバー中に生成したすべてのDAアイテムオブジェクトのオートメーションコレクション。
OPCItem **(DAアイテム)**	アイテム定義、現在値、ステータス、最終更新時刻などを保守するオブジェクト。カスタムインタフェースでは、分離したアイテムオブジェクトを提供していない点に注意してください。

6.3.2 DAサーバオブジェクト

DAサーバーオブジェクトは、データソース(DAカスタムインタフェースサーバー)との交信やデータアクセス(読み出し･書き込み)の方法を提供するもので、グループとアイテムのオブジェクトを作成する前にDAサーバ オートメーションオブジェクトを作成し、次にDAデータアクセスカスタムインタフェースに“connect”メソッドを使って接続します。

DAサーバは次に示すプロパティ、メソッド、イベントを用意しています。

プロパティ	説明
StartTime	DAサーバが起動された時間(UTC)。
CurrentTime	DAサーバで識別する現在時間(UTC)。
LastUpdateTime	このDAクライアントへの最終データ更新時間(UTC)。
MajorVersion	DAサーバソフトウェアのメジャーバージョン。
MinorVersion	DAサーバソフトウェアのマイナーバージョン。
BuildNumber	DAサーバソフトウェアのビルド番号。
VendorInfo	DAサーバについての追加情報として用意されるベンダ独自の文字列。(会社名およびサポートするデバイスのタイプを含めることを推奨します。)
ServerState	DAｻｰﾊﾞｰの実行状態を返す (OPCServerStatusの一部) 。
LocaleID	(R/W) ロケール識別 (言語ID) 文字列を返す。
BandWidth	サーバ固有値。サーバーの利用可能なバンド幅比率 (%) を返す。カスタムサーバーのGeStaus () から得る値。
OPCGroups	DAグループオブジェクトのコレクション。DAサーバーオブジェクトのデフォルトのプロパティ。
PublicGroupNames	サーバーのパブリックグループ名を返す。
ServerName	クライアントが接続したサーバー名を返す。
ServerNode	クライアントが接続したサーバーのノード名を返す。
ClientName	(R/W) クライアント がサーバーに名前を登録する。デバッグ用。

(注) (R/W) は読み出し･書き込みのプロパティをあらわします。注記の無いものは読み出し専用です。

メソッド	説明
GetOPCServers	登録されたDAサーバーの名前 (ProgID) を返す。
Connect	カスタムインタフェースを実装したDA Data Accessサーバに接続する。
Disconnect	DAサーバーから切断する。
CreateBrowser	DAブラウザオブジェクトを生成する。
GetErrorString	エラーコードをロケールIDで指定の文字列に変換する。
QueryAvailableLocaleIDs	サーバー･クライアントセッション間で、利用できるロケールIDを返す。
QueryAvailableProperties	利用できるプロパティを知りたいIアイテムDを指定すると、利用できるプロパティのIDコードリストとデスクリプション (テキスト) を返す。
GetItemProperties	アイテムIDとプロパティIDから、現在のデータ値のリストを返す。
LookUpItemIDs	プロパティIDから該当するアイテムIDのリストを返す。

イベント	説明
ServerShutDown	サーバーがシャットダウンを行うことをクライアントに通知する。クライアントはサーバーと切断できるようこのメソッドを提供する。

6.3.2.1 DAブラウザ オブジェクト

DAブラウザオブジェクトは、サーバー内にあるブランチやアイテムの名前のコレクションです。ブラウジングはオプションですので、サーバーがサポートしてない場合は、CreateBrowserを実行してもこのオブジェクトは生成されません。

プロパティ	説明
Organization	DAサーバのネームスペースのタイプ（フラットまたは階層）。
Filter	（R/W）ShowBranchesやShowLeafsメソッドに適用するフィルタです。このフィルタを使って名前リストの範囲を絞ることができます。
DataType	（R/W）ShowLeafsメソッドに適用する要求データタイプです。
AccessRights	（R/W）ShowLeafsメソッドに適用する要求アクセス権です。
CurrentPosition	ツリー構成における現在位置を返します。
Count	コレクション中の、アイテム数を返します。

（注）（R/W）は読み出し・書き込みのプロパティをあらわします。注記の無いものは読み出し専用です。

メソッド	説明
Item	ItemSpecifier（コレクション中のインデックス）にてインデクスされた名前を返します。
ShowBranches	現在のブラウズポジションのブランチ名をコレクションにいれます。
ShowLeafs	現在のブラウズポジションのリーフ名をコレクションにいれます
MoveUp	階層を1つ上に移動します。
MoveToRoot	第1階層に移動します。
MoveDown	下位の指定ブランチに移動します。
MoveTo	絶対位置に移動します。
GetItemID	アイテムの名前から、アイテムIDを返します。
GetAccessPath	アイテムIDのアクセスパスを文字列で返します。

イベント	説明
ー	

6．3．3　DAグループコレクション（Groups）オブジェクト

DAグループコレクションオブジェクトは、DAグループ オブジェクトのコレクションで、これらの生成、削除、管理を行うオブジェクトです。

プロパティ	説明
Parent	親のDAサーバーへのレファレンスを返します。
DefaultGroupIsActive	(R/W) Groups.Addで生成されたOPCGroupsのために、アクティブステートをデフォルトとします。
DefaultGroupUpdateRate	(R/W) Groups.Addで生成されたOPCGroupsのために、デフォルトのUpDateRateを設定(1000ms)します。
DefaultGroupDeadBand	(R/W) Groups.Addで生成されたOPCGroupsのために、デフォルトのDeadBand(0%)を設定します。
DefaultGroupLacaleID	(R/W) Groups.Addで生成されたOPCGroupsのために、デフォルトのLocaleIDを設定します。
DefaultGroupTimeBias	(R/W) Groups.Addで生成されたOPCGroupsのために、デフォルトのTimeBias(0分)を設定します。
Count	コレクションのプロパティ

(注) (R/W) は読み出し･書き込みのプロパティをあらわします。注記の無いものは読み出し専用です。

メソッド	説明
Item	OPCGroupsのデフォルトメソッド。ItemSpecifier(コレクションのインデックス)で指定されるOPCグループを返します。
Add	新しくDAグループを生成し、コレクションに追加する。
GetOPCGroup	ItemSpecifierのDAグループを返します。
Remove	ItemSpecifierのDAグループを削除します。
RemoveAll	servershutdownに備えて、すべてのDAグループとDAアイテムを削除します。
ConnectPublicGroup	PublicGroupに接続します。
RemovePublicGroup	PublicGroupを削除します。

イベント	説明
AllGroupsDataChange	複数のグループ間にまたがるデータ変化を受け、処理する一つのイベントハンドラの実装用。

6.3.3.1 DAグループ オブジェクト

DAグループは、クライアントが用途に応じて、アクセスしやすくデータを構成する方法を提供するものです。

プロパティ	説明
Parent	親サーバーオブジェクトへのレファレンスを返します。
Name	(R/W)グループの名前の書き込みまたは読み出し。
IsPublic	グループがPublicの時にはTrueを返します。
IsActive	(R/W) グループのアクティブ状態をコントロールします。InActiveなグループはデータの収集を行いません。
IsSubscribed	(R/W)グループへの非同期通知をコントロールします。
ClientHandle	(R/W)グループにつける値。データやステータスとともにクライアントに返されます。
ServerHandle	サーバーがグループに割り当てたユニークな値。
LocaleID	(R/W)ロケールIDの設定、読み出し。
TimeBias	(R/W)データのタイムスタンプをデバイスのローカルタイムに変換するために必用な情報。
DeadBand	(R/W)デッドバンド(フルスケールの%;0-100)を指定します。
UpdateRate	(R/W)アプデートレート(ms単位)を指定します。
OPCItems	OPCIemオブジェクトのコレクション。

(注) (R/W) は読み出し・書き込みのプロパティをあらわします。注記の無いものは読み出し専用です。

メソッド	説明
SyncRead	グループ内アイテムのデータ値、品質フラグ、タイムスタンプを同期読み出しします。データ値は、CacheまたはDeviceから読み出します。
SyncWrite	グループ内アイテムに、サーバーハンドルとデータ値を同期書き込みします。データ値はDeviceに書き込まれます。
AsyncRead	グループ内アイテムのデータ値、品質フラグ、タイムスタンプを非同期読み出しします。結果はASyncReadCompleteイベント経由で返されます。
AsyncWrite	グループ内アイテムに、サーバーハンドル、トランザクションIDとデータ値を非同期書き込みします。結果はAsyncWriteCompleteイベント経由で返されます。
AsyncRefresh	グループ内アクティブな全アイテムのイベントを発生させます。結果はAsyncRefreshCompleteイベント経由で返されます。
AsyncCancel	未処理のトランザクションをキャンセルします。結果はASyncCancelCompleteイベント経由で返されます。

イベント	説明
DataChange	グループ内アイテムのデータ値、またはデータ値の品質フラグに変化が起きたときに発生します。なおグループのUpDateRateより早く発生することはありません。
AsyncRefreshComplete	AsyncRefreshが完了したときに発生ます。
AsyncWriteComplete	AsyncWriteが完了したときに発生します。
AsyncCancelComplete	AsyncCancelが完了した時に発生します。

6．3．3．2 DAアイテムコレクション（Items）オブジェクト

DAアイテムに対するデフォルトのプロパティを有しており、DAアイテムが加えられたとき、下記Defaltxxxが、その初期状態になります。

プロパティ	説明
Parent	親であるDAグループオブジェクトを返します。
DefaultRequestedDataType	(R/W) Addコールを使う時の、要求データタイプのデフォルト値(VT_Empty (cannonical data))。
DefaltAccessPath	(R/W) Addコールを使うときの、アクセスパスのデフォルト値(‘’‘‘)。
DefaultIsActive	(R/W) Addコールを使うときのデフォルトアクティブステート(True)値。
Count	コレクション中のオブジェクト数を返します。

(注) (R/W) は読み出し･書き込みのプロパティをあらわします。注記の無いものは読み出し専用です。

メソッド	説明
Item	ItemSpecifierで指定するOPCItemsコレクション内のDAアイテムを返す。
GetOPCItem	Addの戻り値(サーバハンドル)で指定するDAアイテムを返す。
AddItem	新しいDAアイテムオブジェクトを生成してコレクションに追加する。そのプロパティは、DAアイテムコレクションオブジェクトのデフォルト値で決まります。
Remove	DAアイテムを削除します。
Validate	Addメソッドで正しくDAアイテムが生成されたか否か決める。
SetActive	DAアイテム個々に付いて、アクティブまたは非アクティブに設定する。
SetClientHandles	グループ内アイテムのクライアント ハンドルを設定します。
SetDataTypes	グループ内アイテムの要求データタイプを設定します。

イベント	説明

6．3．3．3 DAアイテムオブジェクト

DAアイテムはサーバー内のデータソースへのコネクションをあらわします。個々のアイテムは、データ値、品質フラグ、タイムスタンプより構成されます。

プロパティ	説明
Parent	親のDAグループオブジェクトへのレファレンスを返します。
ClientHandle	(R/W) DAアイテムにクライアント ハンドルを設定します。このハンドルは、DAグループのイベント発生時にクライアントに返されます。
ServerHandle	サーバーによって割り当てられたハンドルデータを返す。ハンドルデータは個々のDAアイテムをユニークに識別する値。
AccessPath	クライアントが指定したアクセスパスを返す。
AccessRight	アイテムのアクセス権を返す。
ItemID	このアイテムのユニークな識別子を返す。
IsActive	(R/W) このアイテムのデータ収集状態フラグを設定。Trueであればアイテムはアクティブ。
RequestedDataType	(R/W) アイテムのデータ値を返すときの、要求データタイプ。
Value	サーバーから読み出した、最新のデータ値。
Quality	サーバーから読み出した、最新の品質フラグ。
TimeStamp	サーバーから読み出した、最新のタイムスタンプ。
CanonicalDataType	サーバー内のネイティブなデータタイプを返します。
EUType	Engineering Unit (EU) のタイプに関する情報を返します。
EUInfo	Engineering Unit情報を含むVariant値。

(注) (R/W) は読み出し・書き込みのプロパティをあらわします。注記の無いものは読み出し専用です。

メソッド	説明
Read	サーバーからアイテムをブロック読み出しします。
Write	サーバーからアイテムをブロック書き込みします。

イベント	説明
――	

6.4 DAカスタムインタフェース

図6-7～図6-8はDAオブジェクトとインタフェースの要約です。インタフェースのいくつかがオプションであることに注意してください。（オプションのインタフェースは[ ]で示されています。）

なお、インタフェースの名前には次のような取り決めがあります。

IOPCxxx　　OPCカスタムインタフェース

Ixxx　　OLE標準インタフェース(IPersistFile,IDataObjectなど)

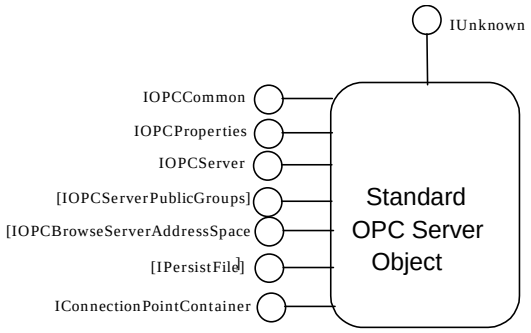


図 6-7 標準（カスタムインタフェース）DAサーバオブジェクト

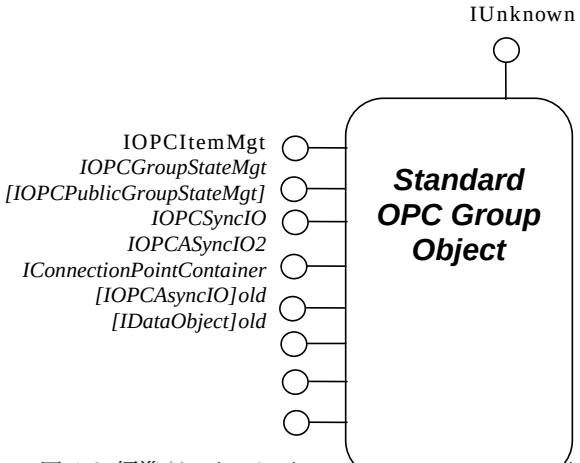


図 6-8 標準（カスタムインタフェース）DAグループオブジェクト

6.4.1 DA カスタムインタフェース オブジェクト

DAカスタムインタフェースには次のカスタムオブジェクトがあります。

- DAサーバオブジェクト
- DAグループオブジェクト

DAカスタムインタフェースは、DAコンポーネントとオブジェクトのインタフェースと動作についてのインタフェースです。DAサーバの開発者はこの節で定義された機能を用意することによりDAオブジェクトを実装することができます[3]。また、この節ではDA準拠のコンポーネントを作成するために必要な標準のOLEインタフェースの動作についても記述しています。

列挙オブジェクトとそのインタフェースについてはOLEの仕様によって十分に定義されているので、ここでは簡単に記述されています。列挙には以下のインタフェースがあります。

- グループの列挙 - (IOPCServer::CreateGroupEnumerator)
- アイテム属性の列挙 - (IOPCItemMgt::CreateEnumerator)
- サーバアドレススペースの列挙 - (IOPCBrowseServerAddressSpace::BrowseOPCItemIDs)
- アクセスパスの列挙 - (IOPCBrowseServerAddressSpace::BrowseAccessPaths)

6.4.2 DAサーバオブジェクト

DAサーバオブジェクトはDAサーバをエクスポーズするメインオブジェクトです。

6.4.2.1 IOPCServerインタフェース

DAサーバのメインインタフェースです。

メンバ	説明
AddGroup	DAサーバへグループを追加します。
GetErrorString	DAサーバのエラーコードに対応するエラー文字列を返します。
GetGroupByName	同一DAクライアントによって作成されたプライベートグループのポインタを返します。(パブリックグループへのアタッチにはGetPublicGroupByNameを使用します。)
GetStatus	DAサーバの現在の状態を取得します。
RemoveGroup	グループを削除します。
CreateGroupEnumerator	DAサーバで用意されたグループの様々な列挙を作成します。

6.4.2.2 IOPCServerPublicGroupsインタフェース(オプション)

パブリックグループを管理するためのインタフェースです。

メンバ	説明
GetPublicGroupByName	DAクライアントをパブリックグループへ接続する。グループへのポインタを返します。
RemovePublicGroup	パブリックグループを削除します。

6.4.2.3 IOPCBrowseServerAddressSpaceインタフェース(オプション)

DAサーバで利用可能なアイテムIDをブラウズするためのインタフェースです。

[3] OLE標準の一部のインタフェースについては省略されています。

メンバ	説明
QueryOrganization	システム構造がフラットか階層構造かを調べます。
ChangeBrowsePosition	階層スペース内で'Up' または'Down'へ移動します。
BrowseOPCItemIDs	アイテムIDのリスト(IEnumString)を返します。(ブラウズの位置はChangeBrowsePositionで設定される。)
GetItemID	階層スペース内のアイテムIDを取得します。
BrowseAccessPaths	アイテムIDのアクセスパス(AccessPaths)をブラウズします。

6．4．2．4 IPersistFileインタフェース(オプション)

DAサーバの構成情報をファイルにセーブしたり、ファイルからロードするためのインタフェースです。

メンバ	説明
IsDirty	構成情報に変更があったかどうかを返します。
Load	構成情報をロードすることをDAサーバに要求します。
Save	構成情報を保存します。
SaveComplete d	スタブとして実装します。
GetCurFile	現在ロードされている構成情報ファイルの名前を返すようにDAサーバに要求します。

6．4．2．5 IOPCItemProperties (new[4])

DAサーバで利用可能なアイテムIDをブラウズするためのインタフェースです。
IOPCBrowseServerAddressSpaceやEnumItemAttributesなどと似た機能ですが、
より簡単に少数のItemIDをブラウズするためのインタフェースです。
通常、あるDA ItemID（ユニーク）には、それに関連するさまざまなプロパティ（値、品質、Hi,Lo.など）が割り当てられます。このプロパティには、その種類ごとにプロパティIDがDAで定められています。

メンバ	説明
QueryAvailableProperties	指定のItemIDに対して、利用できるプロパティIDを返します。
GetItemProperties	クライアントが要求するItemIDとプロパティIDに対して、現在のデータリストを返します。
LookupItemIDs	クライアントが要求するItemIDとプロパティIDに対して、該当するITEMIDリストを返します。

6．4．2．6 IConnectionPointContainer (new)

IDAShutdown（クライアントに実装）とのconnection pointをアクセスするためのインタフェースです。

[4] newは、ver2.0で追加になったインタフェースを示します。

メンバ	説明
EnumConnectionPoints	DAサーバーとクライアント間でサポートされているconnectionpointの列挙を生成します。
FindConnectionPoint	DAサーバーとクライアント間にある特定のconnection pointsを検索します。

6.4.2.7 IOPCCommon（new[5]）

特定のサーバー／クライアントセッションで有効なLocaleIDの設定などを行うインタフェースです。他のOPCｻｰﾊﾞｰ（AlarmandEventなど）と共通に使うことができます。

メンバ	説明
SetLoacleID	デフォルトのLocaleIDを設定します。
GetLocaleID	デフォルトLocaleIDを返します。
QueryAvailableLocaleIDs	利用可能なLocaleIDのリストを返します。
GetErrorString	サーバー固有のエラーコードを返します。
SetClientName	サーバーにクライアントの名前を登録することができます。(デバッグ用)

6.4.3 DAグループ オブジェクト

DAグループオブジェクトはDAサーバがアイテムコレクションの管理をするためのオブジェクトです。

6.4.3.1 IOPCGroupStateMgtインタフェース

グループの全体的な状態を管理します。

メンバ	説明
GetState	グループの現在の状態を取得します。
SetState	グループのプロパティを設定します。
SetName	プライベートグループの名前を変更します。名前はユニークです。また、パブリックグループは変更できません。
CloneGroup	グループのコピーを作成します。

6.4.3.2 IOPCPublicGroupStateMgtインタフェース(オプション)

プライベートグループをパブリックグループに変更するためのオプションのインタフェースです。

メンバ	説明
GetState	グループがパブリックグループかどうか調べます。
MoveToPublic	プライベートグループをパブリックグループに変換します。

6.4.3.3 IOPCSyncIOインタフェース

DAクライアントからDAサーバへの同期読み込み／書き込みを実行します。

[5] NewはVer2.0で追加になったインタフェースをあらわします。

・

メンバ	説明
Read	グループ内のアイテムのデータ値、品質フラグ、タイムスタンプの読み込み。
Write	グループ内のアイテムにデータを書きます。

6．4．3．4 IOPCAsyncIOインタフェース（old[6]）

DAクライアントからDAサーバへの非同期読み込み／書き込みを実行します。

メンバ	説明
Read	グループ内のアイテムを読み込む。
Write	グループ内のアイテムへ書き込む。
Refresh	グループ内のアクティブなアイテムをコールバックします。
Cancel	DAサーバに未処理のトランザクションをキャンセル要求します。

6．4．3．5 IOPCItemMgtインタフェース

グループ内のアイテムの動作を制御します。

メンバ	説明
AddItems	グループへアイテムを追加します。
ValidateItems	アイテムが有効かどうか調べます。アイテムの情報を返します。
RemoveItems	グループからアイテムを削除します。（AddItemsの逆）
SetActiveState	グループ内のアイテムにactive/inactiveを設定します。
SetClientHandles	グループ内のアイテムのクライアントハンドルを変更します。
SetDatatypes	グループ内のアイテムの要求データタイプを変更します。
CreateEnumerator	グループ内のアイテムの列挙を作成します。

6．4．3．6 IEnumOPCItemAttributesインタフェース

グループのアイテムと属性を検索します。

メンバ	説明
Next	グループからアイテムを取得します。
Skip	アイテムをスキップします。
Reset	列挙の先頭アイテムにリセットします。
Clone	列挙の現在状態のコピーを作成します。

6．4．3．7 IDataObjectインタフェース（old）

IDataObjectはDAグループ上で実行されます。

[6] oldは、Ver1.0対応のインタフェースであることを示します。

メンバ	説明
DAdvise	DAグループとDAクライアントの接続を確立します。
DUnadvise	DAグループとDAクライアントの接続を切断します。

6．4．3．8 IOPCASyncIO2インタフェース (new)

IOPCAsyncと同様、クライアントと非同期読み込み／書き込みを行うインタフェースです。クライアントとの接続にはIConnectionPointを使います。

メンバ	説明
Read	DAグループ内のアイテムを読み込みます。
Write	DAグループ内のアイテムに書き込みます。
Refresh2	DAグループ内のアクティブなアイテムをコールバックします。
Cancel2	DAサーバーに未処理のトランザクションのキャンセルを要求します。
SetEnable	OnDataChangeコールバックを制御します。TRUE（デフォルト）/Falseにて、DAグループとのコールバックを有効／無効にすることができます（ただしRefreshには、無関係）。
GetEnable	最後のコールバック制御値（TRUE/FALSE）を復帰します。

6．4．3．9 IConnectionPointContainerインタフェース (new)

ＩＤａｔａＯｂｊｅｃｔに代わるもので、クライアントとのcallback接続を行うインタフェースです。各アイテムではなく、グループに実装します。ver2.0では必須のインタフェースです。

メンバ	説明
EnumConnectionPoints	DAグループとクライアント間のConnectionPointを列挙します。
FindConnectionPoint	DAグループとクライアント間のConnectionPointを検索します。

6．4．4　DAクライアント側インタフェース

6．4．4．1 IAdviseSinkインタフェース（old)

クライアントはOnDataChangeの実装を用意する必要があります。

メンバ	説明
OnDataChange	DAグループからの例外ベースのデータ変化を通知させるためにDAクライアントが用意するメソッドです。

6．4．4．2IOPCDataCallBackインタフェース (new)

ver２ではConnectionPointを使うため、クライアントにはIOPCDataCallBackを実装する必用があります。

メンバ	説明
OnDataChange	DAグループからの例外ベースのデータ変化とリフレシュを通知させるためにDAクライアントが用意するメソッドです。
OnReadComplete	DAグループからの非同期読み込み（AsyncIO2 Read）完了を通知させるためにDAクライアントが用意するメソッドです。
OnWriteComplete	DAグループからの非同期書き込み（AsyncIO2 Write）完了を通知させるためにDAクライアントが用意するメソッドです。
OnCancelComplete	DAグループからの非同期キャンセル要求（AsyncIO2 Cancel）完了を通知させるためにDAクライアントが用意するメソッドです。

6．4．4．3 IOPCShutdownインタフェース（new）

ver2.0対応のクライアントは、IOPCShutdwonを実装するオブジェクトを生成する必要があります。

メンバ	説明
ShutdownRequest	サーバーがshutdownする必要あるときに、サーバーによりコールされます。クライアント はサーバーとのコネクションを切断します。

7 DataAccess Ver1．0とVer2．0の主な違い

7．1 カスタムインタフェース

7．1.1 非同期通信インタフェースの変更

Ver１．０およびＶｅｒ１．０A（以下Ｖｅｒ１．０と記述）[7]では非同期インタフェースにIDataObject(サーバーサイド)とIAdviseSink（クライアントサイド）を適用しましたが、Ver2.0ではより実装しやすく、またスマートにプラグアンドプレイを実現できるIConnectionPointを採用しました。

	サーバー/グループ	クライアント	仕様書上の表記
Ver1.0	IDataObject ::Dadvise ::Dunadvise	IAdviseSink ::OnDataChange	old
Ver2.0	(On Server and Group) IConnectionPointContainer ::FindConnectionPoint ::EnumConnectionPoints	IOPCDataCallback ::OnDataChange ::OnReadComplete ::OnWriteComplete ::OnCancelComplete	new

これに伴い、下記非同期インタフェースの仕様、例外処理も変更になりました。

7．1.2 非同期インタフェースの名称

上記１の理由により、DAでの非同期インタフェースの名称とメソッドとそのパラメータなどが変更になりました。
Ver２では、IOPCAsyncIO２を使用することが必須条件です。また、SetEnable、GetEnableというメソッドが追加になりOnDataChangeの動作を制御できるようになりました。

バージョン	インタフェース	メソッド
Ver1.0	IOPCAsync	Read,Write,Cancel、Refresh
Ver2.0	IOPCAsyncIO2	Read,Write、Cancel2,Refresh2、SetEnable、GetEnable

7．1.3 非同期インタフェースの動作

DAサーバとDAクライアント間で同期あるいは非同期データ転送を行う際、グループ、アイテムのアクティブフラグの状態が、DAサーバの挙動に影響します。Ver２．０ではCallBack::OnDataChangeを制御するSetStateが追加されており、特にsubscription（変化通知）での挙動に若干の相違があります。SubscriptionとはUpDateRateで指定した周期で、デバイスのデータとQuality（品質）データを取得・更新するものですが、Ver２．０ではcallbackがTrueであれば、Qualityの変化をみて、新しいデータとQualtyを非同期通信路（callback）を介して、クライアントに転送し、サーバー自身のキャシュの更新を行いますが、Falseであればサーバーは自身のキャッシュの更新のみとなります。ver1.0ではこの切り替えができなく、サーバーのキャッシュ更新とデバイスデータの転送を非同期通信路（IAdviseSink）を介しておこないます。

---

[7] OPC DataAccess Specification Ver１．０ とVer１．０Aでは、機能的にほぼ同等のため、Ver１．０で代表して記載します。

7．1.4 追加されたインタフェースと機能

下記インタフェースおよび機能がVer２．０であらたに追加になりました。

追加機能またはインタフェース	主な機能	備考
IOPCProperties ::QueryAvailableProperties ::GetIytemProperties ::LookupItemIDs	クライアントがアイテムの属性やtagなどをより簡単にアクセスできるようにするための“便利”インタフェース。ItemIDを直接指定する代わりに、DataTyteなどで分類したPropertyIDを指定することができる。	PropertyIDSetはDAで定義される。
IOPCCommon ::SetLocaleID ::GetLocaleID ::QueryAvailableLocaleIDs ::GetErrorString ::SetClientName	特定のサーバ／クライアント間でLocaleIDの設定、問い合わせをする機能を提供するもの。	他のOPCサーバ（A&Eなど）でも実装する。
BrowseServerAdddressSpace::ChangeBrowsePosition	BrowsingDirection OPC_BROWSE_TOが追加された。	Ver１．０ではUP/DOWNのみ。
Shutdown機能 IOPCShutdown:: ShutdownRequest（クライアント用） IConnenctionPointContainer（サーバ上）	DAサーバーがクラッシュなどの異常になったときに、そのサーバーが接続しているクライアントを探せるようにするもの。クライアントはサーバーのshutdown要求と理由を受け、サーバーとのコネクションを開放できる。	クライアントが複数のサーバーとコネクトしても可能。
グループの列挙方法： IEnumUnkownが削除された。	IOPCServer::CreateGroupEnumratorを使う	実装上の問題。

7．1.5 バージョンによる互換性一覧表

下記にVer1.0とVer2.0の互換表を示します。
現状Ver1.0とVer2.0対応のDAサーバとクライアントが混在する可能性がありますが、
サーバーベンダに対しては、既存のｻｰﾊﾞｰにはVer２．０の機能の追加ならびに、既存のVer1.0対応のクライアントのためにVer1.0のサポート機能を残すことが推奨されています。

Data Access Server **Required Interfaces**	**1.0**	**2.0**
OPCServer		
Iunknown	Required	Required
IOPCServer	Required	Required
IOPCCommon	N/A	Required
IConnectionPointContainer	N/A	Required
IOPCItemProperties	N/A	Required
IOPCServerPublicGroups	Optional	Optional
IOPCBrowseServerAddressSpace	Optional	Optional

OPCGroup		
Iunknown	Required	Required
IOPCItemMgt	Required	Required
IOPCGroupStateMgt	Required	Required
IOPCPublicGroupStateMgt	Optional	Optional
IOPCSyncIO	Required	Required
IOPCAsyncIO2	N/A	Required
IconnectionPointContainer	N/A	Required
IOPCAsyncIO	Required	N/A
IdataObject	Required	N/A

7.2オートメーションインタフェース

7.2.1 主な相違点

下表に主な相違点を記載します。

項目	Ver1.0	Ver2.0
仕様書	カスタムインタフェースVer1.0A仕様書と同一仕様書。	カスタムインタフェースVer2.0仕様書とは別冊。
サポート言語	VisualBasic ver4、 VBA４	VisualBasic ver５、６ VBA５，６
インタフェース実装ならびに表記方法	デュアルインタフェースとしてカスタムインタフェースとともに実装。カスタムインタフェースと区別するため、IOPCxxxDispと表記。	ラッパDLLを経由して、カスタムインタフェースVer2.0をアクセスする。オートメーションクライアントは、サーバのカスタムインタフェースを意識する必要はない。
オブジェクトモデル	コレクションオブジェクトのモデルはなし。	コレクションオブジェクトモデルとして、OPCGroups,OPCItemsを追加。
	OPCBrowerオブジェクトなし。	DAブラウザオブジェクトモデルを追加。
イベントプロパティ	サポートなし。	VB５のイベント機能をサポート。
カスタムインタフェースver２．０新インタフェースのサポート	なし	サポートする。

7.2.2 互換性

オートメーションｖｅｒ１．０とｖｅｒ２．０では、ＶＢ Ｓｙｎｔａｘにかなりの相違があるため、コードレベルでの互換性はありません。

7．3　データ読み込みとアクティブフラグの関係

7．3．1 同期読み込み（IOPCSyncIO：Ver1.0及びVer2.0）

IOPCSyncIO::Read					
Source	Enable Callbacks（＊）	Group Active State	Item Active State	読み込み動作 ○：正しくデータを読める。 ×：データを読めない、または品質フラグにエラービットがセットされる	備考
Cache	NA	Active	Active	○ （値と品質）	
Cache	NA	Active	InActive	× （OUT_OF_SERVICEの品質フラグ）	
Cache	NA	InActive	NA	× （OUT_OF_SERVICEの品質フラグ）	
Device	NA	NA	NA	○ （値と品質フラグが返る。サーバはキャッシュを更新する）	

（＊）AsyncIO2::SetEnable、DisEnableによる（Ver2.0）。

7．3．2 非同期読み込み（IOPCAsyncIO2）

IOPCAsyncIO2::Read					
Source	Enable Callbacks	Group Active State	Item Active State	読み込み動作	備考
NA	NA	NA	NA	○ （DEVICEから読みこんだ値と品質フラグを得る。サーバは、キャッシュの値と品質フラグを更新）	

7．3．3 リフレッシュ

IOPCAsyncIO2::Refresh2					
Source	Enable Callbacks	Group Active State	Item Active State	読み込み動作	備考
Cache	NA	Active	Active	○ （キャシュ内の値と品質フラグを得る。	.
Cache	NA	Active	InActive	× （グループ内の非アクティブなアイテムのデータは、送信されない。アクティブなアイテムがゼロのときは、エラーメッセージを返す。）	
Cache	NA	InActive	NA	× （E_FAILを返す。）	
Device	NA	Active	Active	○ （DEVICEからの値と品質フラグを返す。キャッシュも更新する。）	
Device	NA	Active	InActive	×	
Device	NA	InActive	NA	×	

7．3．4 サブスクリプション（IOPCDataCallback経由のばあい）

Subscription via IOPCDataCallback::OnDataChange					
Source	Enable Callbacks	Group Active State	Item Active State	読み込み動作	備考
NA	TRUE	Active	Active	○ 更新周期でスキャンして、DEVICEからよみとった値と品質フラグを送信する。 品質フラグが変化した場合も同様。 品質が変化ないとき、デッドバンドとの比較をし、変化幅を超えるときも、送信する。	
	TRUE	Active	InActive	× 物理データソースから、値を収集するだけ。	.
	TRUE	InActive	NA	× 物理データソースから、値を収集するだけ。	
NA	FALSE	Active	Active	× 更新周期でスキャンして、DEVICEからよみとった値と品質フラグを送信する。品質フラグが変化した場合も同様。 品質が変化ないとき、デッドバンドとの比較をし、変化幅を超えるとき、送信はせずにキャシュの更新のみ行う。	
NA	FALSE	Active	InActive	× サーバーはアクティブなアイテムの物理データソースから、データを収集はするが、送信はしない。	
NA	FALSE	InActive	NA	× （同上）	

7．3．5 非同期読み込み（IOPCASyncIO ／ver1.0）

IOPCAsyncIO::Read (through the IadviseSink::OnDataChange method)					
Source	Enable Callbacks	Group Active State	Item Active State	読み込み動作	備考
Cache	NA	Active	Active	○ キャシュ内の値と品質フラグが送信される。	
Cache	NA	Active	InActive	× 品質フラグに、out_of__serviceがセットされ、送信される。	
Cache	NA	InActive	NA	× （同上）	
Device	NA	NA	NA	○ DEVICEの値と品質フラグが送信される。キャッシュも更新される。	

リフレッシュ（Ver1.0）

IOPCAsyncIO::Refresh					
Source	Enable Callbacks	Group Active State	Item Active State	読み込み動作	備考
Cache	NA	Active	Active	○ 値と品質フラグが送信される。	
Cache	NA	Active	InActive	× 非アクティブなアイテムの値、品質フラグは送信されない。	
Cache	NA	InActive	NA	× E_FAILがリターンされる。	.
Device	NA	Active	Active	○ DEVICEから読み込んだ値と品質フラグを送信する。同時にキャッシュも更新する。	
Device	NA	Active	InActive	×	
Device	NA	InActive	NA	× E_FAILをリターンする。	

7．3．6 サブスクリプション（IDataObject (old)経由）

Subscription via (IDataObject::DAdvise) & (IAdviseSink::OnDataChange)					
S Source	Enable Callbacks	Group Active State	Item Active State	読み込み動作	備考
NA	NA	Active	Active	○ 更新周期でスキャンして、DEVICEからよみとった値と品質フラグを送信する。 品質フラグが変化した場合も同様。 品質が変化ないとき､デッドバンドとの比較をし、変化幅を超えるときも、送信する。	
NA	NA	Active	InActive	× 物理デバイスのデータを収集するのみ。	
NA	NA	InActive	NA	× 同上	

8 OPC共通仕様

上述の3種類のＯＰＣ仕様（DataAccess、Historical DataAccess、Alarm and Events）は、互いに独立した仕様ではありますが、共通している機能も多く、”OPC Common”として、規定されている事項を本項で、記載します。

8．1 カスタムインタフェースとwrapper dll

OPCサーバのカスタムインタースは、必須で、OPCサーバベンダは必ず実装する必要がありますが、オートメーションインタフェースに関しては、wrapper dllが各OPCの仕様ごとにOPCーFから提供されますので、これを利用することができます。

8．2 共通インタフェース

下記インタフェースは、各OPC仕様のサーバーに共通のインタフェースです。すなわち、OPCData Access Server、Alarm and Events Server、Histrical Data Access Server のいずれにも実装する必要があります。

（１）OPCShutdown機能

サーバーがクラッシュなど異常な状態に成ったときなど、接続しているクライアントに接続解除を通知するためのインタフェースです。クライアントは、IOPCShutdownとIUnkownインタフェースをもったオブジェクトを接続するOPCサーバーごとに生成する必要があります。OPCサーバには、このIOPCShutdownインタフェースを持つオブジェクトを列挙し、メッセージを通知するためのインタフェースIConnectionPointを実装します。

（２）IOPCCommon

特定のクライアントとサーバーセッションで有効（他のクライアントに影響を与えない）となる言語ＩＤの設定や問い合わせを行うために、サーバーに実装する必要があります。

詳細は、“ 7.1.4追加されたインタフェースと機能 ”を参照ください。

8．3 コンポーネントカテゴリとOPCサーバブラウザ

8．3．1コンポーネントカテゴリ

OPCサーバの実装の際、Data Access Ver1.0の段階ではコンポネントカテゴリを使用するという考えはありませんでしたが、現在OPCサーバのカテゴリ（DataAccessやAlarm＆Events、Historical Data Accessなど）とバージョンごとに、コンポーネントカテゴリを生成する必要があります。このコンポーネントカテゴリを適用することで、多くのOPC コンポーネントをマシン上に実装した場合でも、その管理ならびにクライアント からOPCサーバを閲覧する際、効率的に実現することができます。
なお、 さまざまな種類のOPCサーバに対応するために、このコンポーネントカテゴリのタイプは、OPC仕様ごとに定義されて（opc_cat.cファイル）います。
OPC DataAccessの場合、下記のようにバージョンがわかるようなCATIDとそのGUID番号を定義しています。サーバの実装に当たっては、まずこれらのコンポーネントカテゴリを生成し、次にそのコンポーネントカテゴリにサーバを登録します。

（例）

"OPC Data Access Servers Version 1.0"

CATID_OPCDAServer10 = {63D5F430-CFE4-11d1-B2C8-0060083BA1FB}

"OPC Data Access Servers Version 2.0"

CATID_OPCDAServer20 = {63D5F432-CFE4-11d1-B2C8-0060083BA1FB}

8.3.2 OPCサーバブラウザ

ローカルまたはリモートマシン上に、どのようなOPCサーバが実装されているかを、クライアントが調べるために、上記コンポーネントカテゴリと、OPCから提供されるOPC Server Browser（OPCENUM.EXE）ならびにマーシャリングＤＬＬ（OPCCommon_ps.dll）を使用します。

OPCENUM.EXEは、OPCサーバを実装しているマシン（ローカルまたはリモートマシン）に実装します。OPCクライアントは、OPCサーバが実装されているマシン内にOPCENUM.EXEが実装されていれば、下記IOPCServerListというインタフェースとメソッドを使い、ローカルまたはリモートにあるマシンの各コンポーネントカテゴリを調べ、そこに登録されているOPCサーバの詳細を閲覧することができます。

表8.1 IOPCServerList

	IOPCServerList	機能
メソッド	::EnumClassesofCategory	ターゲットマシンに対して、ＣＡＴＩＤを指定し、そのＣＬＳＩＤ（=ＧＵＩＤ番号）を列挙する。
	::GetClassDetails	ＣＬＳＩＤを指定して、そこからＰｒｏｇＩＤやユーザが読めるサーバの名前を得る。
	::CLSIDFromProgID	ＰｒｏｇＩＤを指定して、そこからＣＬＳＩＤ（＝ＧＵＩＤ番号）を得る。

9　技術基盤説明

OPC自体はそれほど特長のあるものではありませんが、OPCの特長の多くは、それがOLE/COM技術の上に構築されていることに起因しています。したがって、OLE/COMを理解することはOPCを理解する早道でもあります。

ここでは、マイクロソフトのオブジェクト技術であるOLE/COMについて概略理解し、さらなる情報へのポインタを示すことを目的として、説明を行っています。

注意：以下で参照する情報は、複数の情報源のうち最も適切でかつ詳細な情報だけを選択しています。

- オブジェクトモデル

COMのオブジェクトモデルは、アプリケーションをCOMオブジェクトの集合として構築した場合に、そのオブジェクト構造が外部アプリケーションからどのように見えるかを示している。狭義にはCOMサーバの階層的なオブジェクト構造を指す場合が多い。オブジェクトモデルは、必ずしもアプリケーションの実装上の構造とは一致しない。たとえば、C++でアプリケーションを構築した場合、C++のクラス定義がCOMオブジェクトに1対1に対応するわけではない。オブジェクトモデルをどのように設計するかでアプリケーション間連携の性能や機能の使い勝手が決まってしまうので、設計は重要な問題である。OPCでは、すでにOPCアプリケーションのオブジェクトモデルは決められているので、開発者が設計の必要はない。よりわかりやすいオブジェクトモデルの一般的説明は、**http://www.microsoft.com/oledev/oleauto/ole2soln.htm**に書かれている。

- COMオブジェクト生成

COM オブジェクト生成は C++では CoCreateInstance、CoGetClassObject などの API、Visual Basic では、CreateObject、GetObject の API で生成される。これにより、オブジェクトのインタフェースのポインタがローカル変数へ返されるので、それを用いてプロパティやメソッドへアクセスする。オブジェクト生成には、そのクラス識別子を指定する必要があるが、一旦生成されると、そのインタフェースポインタがどのオブジェクトのものかを考えてプログラムする必要はなくなる。これらとは別に、モニカーと呼ばれるものや、HTMLの<OBJECT>タグでオブジェクトを生成することも可能である。より詳しいオブジェクト生成の説明は、 **OLE KBase Q104139** および **Microsoft Systems Journal 1996 May, Introducing Distributed COM and the New OLE Features in Windows NT 4.0** に書かれている。

- クラスファクトリ

オブジェクトの機能をクライアントが利用できるようにするためには、オブジェクトを作り、そのインタフェースをクライアントへ返す機能をサーバが提供しなければならない。サーバとはオブジェクトを提供するアプリケーション、クライアントはそのオブジェクトを利用するアプリケーションである。サーバのオブジェクトを作り出す機能、クラスファクトリをシステムに登録し、クライアントからの要求がシステムからサーバに導かれるようにする。クラスファクトリの登録方法は、DLLサーバとEXEサーバでは異なる。クラスファクトリの作成方法はサーバの開発者が知らなければいけない基本知識であり、**Component Object Model(COM) Specification 0.9 Chapter 6, 6.2 Implementing the Class Factory, 6.3 Exposing the Class Factory**に書かれている。

- コレクションオブジェクト

コレクションオブジェクトとは、オブジェクトの集合を扱うため、集合への追加、削除、要素検索、要素総数、などを容易に扱えるようなメソッドやプロパティをもつ集合オブジェクトである。配列やリスト構造のようなもので実装される。実装に関する詳しい説明は、**http://www.microsoft.com/oledev/oleauto/collect.htm (MFC), OLE KBase Q107546** に書かれており、OPC の OPCGroup オブジェクトの実装などサーバ開発で必要とされる。

- オートメーションインタフェース

クライアントがサーバのインタフェースを実行時に選択してアクセスする動的バインディングと呼ばれる機能。これに対して、開発時にサーバのインタフェースをコード上に固定的に書き込むのがVTABLEインタフェースに基づく静的バインディングと呼ばれるものである。オートメーションでは、インタフェース名の一致からサーバのインタフェースが選択され、実行される。サーバにおけるオートメーションの実装方法は、**Inside OLE 2nd Edition, Chapter 14 Five Variations on the Theme of Implementing a Simple Automation Object**に書かれている。クライアントからサーバへのオートメーションの利用を説明した例は、Visual Basicは**Visual Basic Programmer's Guide, Chapter 9 Programming Other Applications' Objects**、Visual C++は、**Microsoft Systems Journal 1995 June OLE Q&A, MSDN Technical Articles/Windows Articles/OLE Articles/Component Object Model, MFC/COM Objects 7: Creating and Using COM Objects with OLE Automation Interfaces**にある。

- カスタムインタフェース

OLEが標準で定義したインタフェース以外の、個別拡張インタフェースをカスタムインタフェースと呼ぶ。カスタムインタフェースはクライアントからVTABLE経由で静的バインディングでアクセスされるインタフェースである。カスタムインタフェースをEXEサーバが実装する場合、インタフェースごとのプロキシコード、スタブコードと呼ばれるDLLを同時に開発し、インストールする必要がある。プロキシとスタブコードの説明は、**Component Object Model(COM) Specification 0.9 Chapter 7 Interface Remoting**にあり、その開発はカスタムインタフェースの開発の例として、**http://www.microsoft.com/msdn/sdk/platforms/doc/activex/src/custintf_2.htm**以降の**ActiveX SDK**および**http://www.microsoft.com/msdn/library/mfccom.htmのTom's Handy Dandy MFC/COM/MIDL Recipe Book for Creating Custom Interfaces**に書かれている。

- Dualインタフェース

Dualインタフェースとは、オートメーションインタフェースの動的バインディングの機能にカスタムインタフェースの静的バインディングを組み合わせ、どちらの方法でもクライアントからオブジェクト機能にアクセスできるようにするための技術である。その説明と実装方法は、**Microsoft Systems Journal 1995 Volume 10 December 12 OLE Q&A, Visual C++ MFC Technical Notes TN065 (MFC)**に書かれている。OPCではカスタムインタフェースとDualインタフェースをサポートするオートメーションインタフェースを個別に定義している。

- In-Proc Handler(in-processハンドラ)

クライアントがサーバと連携するとき、連携の性能向上、セキュリティや信頼性の強化などの目的に必要とされる技術である。したがって、必ずしも連携のための必須機能ではない。in-processハンドラはクライアントマシンのクライアントプログラムと協調するDLL形式のプログラムで、そのプログラムが擬似的なサーバオブジェクトを生成して、クライアントに対して真のサーバオブジェクトの“ふり”をする形で実装される。in-processハンドラは、通常のDLL形式のCOMサーバと実装上は変わらない。したがって、その実装方法はDLLサーバの実装方法、例えば、 **http://www.microsoft.com/intdev/sdk/DLLSERVE-Lesson 7**を参照するとよい。

- イベント通知

クライアントがサーバへ非同期要求を行ったとき、あるいは、サーバやネットワークエラーが発生したとき、その結果通知をイベントとしてクライアントは受信する。クライアントはあらかじめ受信するイベントの種類と、イベント通知で呼ばれる処理をサーバに登録しておく必要がある。イベント通知にはOLEの標準インタフェース、IAdviseSinkを利用することがOPCでは決められている。その開発方法は、**Inside OLE 2nd Edition, Chapter 10 Uniform Data Transfer and Notifications**などに書かれている。

- スレッディングモデル

COM/DCOMにはオブジェクトをマルチスレッド環境で実行させる方法の規定がある。これをスレッディングモデルと呼ぶ。オブジェクトの開発者は、実行に使うスレッディングモデルをシステムに登録し（DLLサーバではレジストリに、EXEサーバではCoInitializeEx()の引数に設定）、そのモデルに合った実装を行う必要がある。スレッディングモデルの種類は、**OLE KBase Q150777**に書かれている。その実装のサンプルとして、OPCのサーバオブジェクトに適切なフリースレッドモデルの例が、**http://www.microsoft.com/intdev/sdk/ FRESERVE-Lesson 17** にある。

- 管理ツール

✓ OLEView－システムにインストールされたCOMオブジェクト、そのインタフェース、各種レジストリ情報を一覧する。
✓ DCOMCNFG－DCOMオブジェクトの起動方法、セキュリティ、その他の管理をするNT4.0標準のGUIツール

- デバッグツール

✓ HookOLE－OLE、COM呼び出しのトレースを行う
✓ ROTView－Running Object Tableに登録されたCOMオブジェクトのチェック
✓ NuMega社のBoundsChecker 4.2

10 付録

10. 1用語集

アルファベット順

A

◆ ActiveX コントロール
ActiveX コントロールはコンテナと呼ばれるアプリケーションで利用してソフト製品を開発します。コンポーネント間の連携をスクリプト言語で記述することで、カスタマイズ可能です。ActiveX の技術により、コンポーネントを利用時にインターネットからダウンロードして実行することもできます。

◆ Application Programming Interface (API)
OS がアプリケーションに対して公開しているプログラムインタフェースです。アプリケーションは、API を経由して OS の機能を利用します。現在、一般的な OS の API は関数の形式をとっており、アプリケーションからは適当なパラメータ(引数)を指定して API の関数を呼び出します。

◆ ATL (Active Template Library)
ATL は、MS VC++ 5.0 に付属するテンプレートを基本にした C++ のクラスの集合です。このテンプレートを使うと、小さくて高速な COM (Component Object Model) オブジェクトを簡単に作成できます。ATL は COM の主要な機能をサポートしています。その中には、IUnknown、IClassFactory、IClassFactory2、そして IDispatch のインプリメンテーションと、デュアル インターフェイス、標準の COM 列挙子 インターフェイス、コネクション ポイント、ティアオフ インターフェイス、そして ActiveX コントロールが含まれます。
ATL コードを使って、シングルスレッド オブジェクト、アパートメント モデル オブジェクト、フリースレッド モデル オブジェクト、またはフリースレッドのアパートメント モデルのオブジェクトを作成できます。

C

◆ CLSID
CLSID はレジストリの登録されたキーの 1 種でコンポーネントの名前や場所の情報をサブキーとして持ち、コンポーネントを含む DLL を特定する目的で利用されます。

◆ COM (Component Object Model)
ソフトコンポーネントの持つべき基本サービスを提供するアーキテクチャです。オブジェクトの生成と消去、バージョン管理、インタフェースの実行時折衝、メモリ管理、ネットワーク透過性などを提供します。ActiveX コントロールは、OLE 複合ドキュメントなどのオブジェクト間連携技術です。DCOM 欄参照。

◆ COM+ (Component Object Model +)
DCOM Version 2.0 のことです.

D

◆ DCOM (Distributed COM、分散 COM)
COM のオブジェクト間連携機能をネットワークのクライアント／サーバへ拡張したものです。Windows NT 4.0 標準装備です。COM の欄参照。

◆ DDE (Dynamic Data Exchange)
Windows に依存したアプリケーション間通信機構。初期(Windows3.0)の OLE は DDE で実現されていました。

◆ Dynamic Link Library (DLL)

Windows プログラム（Windows カーネルなどのシステムプログラム、アプリケーション）は、通常の.EXE ファイルとは別にダイナミックリンクライブラリを用意し、これを実行時にリンクすることができます。 実際には Windows カーネル（KERNEL、USER、GDI）やデバイスドライバもダイナミックリンクライブラリです。複数のプログラムで 1 つの.DLL ファイルを共有できるので、ディスクサイズや実行時メモリを節約することができます。

◆ DCS
Distributed Control System の略です。

E

◆ Embedded Object
エンベディデド オブジェクトとは、OLE 複合ドキュメントにそのデータごと完全に埋め込まれたオブジェクトを指します。

◆ EXE Server
EXE サーバはコンテナのプロセスとは別のプロセスとして動作する EXE タイプの OLE サーバのことです。 アウトプロセスサーバーのうちコンテナと同じマシンで実行されるものがローカルサーバーであり、マシン境界を越えて実行されるものがリモートサーバです。
In-Proc Server / Local Server / Remote Server 参照

F

◆ FILETIME
FILETIME 構造は、ファイルの 64 ビット長の時間値を持ちます。この値は、1601 年 1 月 1 日からの通算時間を 100 ナノ秒の単位で表現します。

G

◆ GUID (Globally Unique Identifier)
OLE コンポーネントオブジェクトに割り振られる ID であり、特殊なアルゴリズムにより生成されます。 分散システム上に無数のオブジェクトが存在するような状況においても、名前の競合が起こらない環境を実現するユニークな ID をこのアルゴリズムは全てのコンポーネントに対して提供します。 OLE2 SDK で提供されるソフトウェアを使って、ソフトウェアデベロッパーはコンポーネントオブジェクトに対する GUID を簡単に作成することができます。

I

◆ IDispatch インタフェース
OLE オートメーションインタフェースが備えるべきインタフェースで、クライアントは通常、このインタフェースを通してメソッドやプロパティにアクセスします。

◆ In-Proc Server
インプロセスサーバはコンテナのプロセス内で動作する、DLL タイプの OLE サーバのことです。 具体的には OCX のことを示します。これまでの OLE サーバがユーザやオートメーションプログラムからのトリガーによりメモリ上にプロセスを作成したのに対して、DLL である OCX は常にメモリ上に展開されています。
EXE Server / Local Server / Remote Server 参照

◆ Interface Definition Language (IDL)
オブジェクトを実装するとき、実装とインタフェースを分離することが実装のカプセル化には重要です。そのために、実装言語とは独立する中間言語でインタフェースを定義し、それをターゲットの実装言語にマップするようにします。こうすることで、オブジェクトの実装は言語非依存となります。IDL が表現する COM オブジェクトの属性は、インタフェースの引数のデータ型や並び順、戻り値、インタフェース識別子、クラス識別子、各国語情報（locale ID）などです。IDL のソースファイルはコンパイルされ、ヘッダファイル、プロキシ、スタブのソースコードへ変換され、コンパイルされて、オブジェクトやクライアントの実装コードとリンクされます。

L

◆ LCID
LCID 属性は、タイプライブラリあるいは関数 argument 用のロカールを確認します。

◆ Linked Object
リンクト オブジェクトとは、OLE で複合ドキュメントに連結されたオブジェクトを指します。

◆ Local Server
Local Server は同一マシン上の独立したプロセス空間で走る OLE サーバアプリケーションです。この場合、クライアントと Local Server は別プロセスとして実行されるため、信頼性は向上しますが、性能は In-Proc Server に対してかなり遅くなります。したがって、サーバの設計ではどちらの方法で実装するほうがいいかの選択をする必要があります。
EXE Server / In-Proc Server / Remote Server 参照

M

◆ MES
Manufacturing Execution System の略です。

◆ Microsoft Foundation Classes (MFC)
マイクロソフトの Windows（Windows 95）、Windows NT アプリケーション開発用の C++クラスライブラリです。マイクロソフトが販売する C++処理系パッケージである Visual C++に付属しています。

◆ MIDL 3.0
IDL をコンパイルしてヘッダ、プロキシとスタブのソースコードを出力するためのツールです。MIDL は OpenGroup の DCE RPC と上位互換機能を持ち、また、MIDL3.0 は従来の ODL ファイルをコンパイルする MKTYPLIB ツールを代替する機能を持ちます。

O

◆ OCX(OLE Custom Control)
OCX はコンポーネントオブジェクトおよびオートメーションオブジェクトの一種です。OCX は複数のソフトウェアベンダから提供され、DB アクセスやマルチメディア、グループウェア機能などのソフトウェア部品を構成します。OLE による複合ドキュメント作成のケースと同じ方式で、OCX を利用した“複合アプリケーション”の開発を行うことができます。ただし、OCX はコンテナのプロセス内で動作する In-Proc Server であるため、通常の OLE サーバとは異なり瞬時に反応することができます。

◆ OLE (Object Linking & Embedding)
マイクロソフトが開発したオブジェクト連携技術です。 COM のベースアーキテクチャの上で提供される種々のアプリケーション機能です。複合ドキュメント、ドラッグ＆ドロップ、オブジェクト保存、OLE オートメーション、OLE コントロールなどが存在します。そのうち、ソフトウエア再利用技術と分散処理技術が ActiveX 技術として改称され、拡張されました。

◆ OLE オートメーション
外部のプログラムから既存のアプリケーション機能を利用するための技術です。ActiveX コントロールが主に GUI を持つ機能に向くのに対して、オートメーションは計算処理など負荷が大きなサーバー型のアプリケーションに適用されます。スクリプト言語（例：Visual Basic）により機能の制御が可能です。
クライアント（OLE オートメーションコントローラ）とサーバ（OLE オートメーションサーバ）の間のオブジェクト連携を可能にしています。

◆ OPC-F (OPC Foundation)
米国に本拠を置く国際非営利法人です。

P

◆ PLC
Programmable Logic Controller の略です。

◆ ProgID
ProgID はレジストリの登録されたキーの 1 種でコンポーネントを識別する文字列を CLSID に変換する。Visual Basic など、コンポーネントを CLSID ではなく ProgID で識別する言語もあります。

◆ Proxy
Proxy はクライアントアプリケーションと同一のアドレス空間に存在するライブラリです。クライアントがリモートオブジェクトと通信するとき、メソッド呼び出しやプロパティへのアクセスを RPC メッセージの正規表現に変換するためのコードです。Proxy はクライアントプログラムの実行時に自動的にロードされます。

R

◆ Remote Procedure Call (RPC)
RPC は分散したプログラム同士が通信し合うための基本機能です。 各プログラムの入出力インタフェースをそれぞれ IDL（インタフェース定義言語）という統一した言語で記述します。

◆ Remote Server
Remote Server はクライアントアプリケーションとは別のマシン上で DCOM を介して実行される OLE サーバアプリケーションです。EXE Server /In-Proc Server/Local Server 参照

S

◆ SCADA
Supervisory Control and Data Acquisition の略です。

◆ Software Development Kit (SDK)
Windows アプリケーションプログラマ向けのソフトウェア開発キットです。マイクロソフトが自社の OS 向けのアプリケーション開発環境をセットにしたものをこう呼んでいます。SDK には、各種サンプルプログラムやヘッダファイル、ライブラリ（スタティックライブラリ／ダイナミックリンクライブラリ／インポートライブラリ）、ドキュメントなど、言語処理系（コンパイラ）を除く関連ファイルがひとまとめにされています。

◆ Stub
サーバ側のオブジェクトに RPC メッセージを渡すときに、正規表現からローカル表現に変換するためのコードです。さらに、オブジェクト実行中のサーバのメモリからのアンロードを抑制するロック機能も果たします。

T

◆ TCP/IP (Transmission Control Protocol / Internet Protocol)
現在のインターネットを支えるネットワークプロトコルです。米国の研究機関や大学を接続した ARPANET 用に開発されたプロトコルです。ARPANET はのちインターネットの源流の一つになりました。

U

◆ URL (Uniform Resource Locator)
ネットワークにどのようなプロトコルでアクセスするかを明示的に指定するための書式です。http や ftp など URL の先頭に指定されたプロトコルに応じてアクセス用のアプリケーションを使い分けます。

◆ UNICODE
UNICODE は米国のコンピュータメーカ団体である UNICODE コンソーシアムによって規格化された、業界標準の統一文字コード国際規格です。

◆ UTC
Windows はシステム時間のコーディネートされた普遍的な時間（UTC）の基礎を形成します。UTC 時間はほぼグリニッジ（イングランド）の現在の日付と時間として定義されています。

◆ UDT(Uniform Data Transfer)
統一データ転送は、オブジェクト間の標準的な情報交換の方法を提供するテクノロジーです。UDT は DDE の後継的位置づけにあります。

V

◆ Variant
OLE オートメーション互換データ型ともいわれ、Visual Basic 変数のデフォルトデータ型としても知られる自己記述データ型です。

◆ Visual Basic(VB)
マイクロソフトの簡易版言語開発ツールです。

◆ Visual Basic for Applications(VBA)
マイクロソフト社が自社のアプリケーションのための共通マクロ言語として採用しようとしているプログラミング言語です。基本的な言語仕様は Visual Basic に準拠しています。VBA はマイクロソフト オフィスを中心とするアプリケーションプログラミング用の Visual Basic のサブセットとして位置づけられ、単体でのパッケージ販売はされません。

◆ Visual C++
マイクロソフトの C 言語開発ツールです。

◆ VTABLE
COM インタフェースを表現するメモリ上構造。これをオブジェクト間の連携のバイナリ表現として標準化したのが COM の基本仕様です。VTABLE は C++のネストクラスやインタフェースクラスの多重継承で実装されるが、C や Java などの言語でも同様の構造体を作成して実装可能です。

五十音順

イ

◆ インターネット
企業や学術施設、商用ネットワークを相互に結んだ世界最大のネットワークです。電子メール、ファイル転送、リモート接続、WWW など、TCP/IP をサポートする様々なサービスが提供されています。利用者は現在世界 150 ケ国以上、1 億人と言われています。

◆ イントラネット
インターネットの仕組みをそのまま、企業や団体などの組織内に構築したものです。社内ネットワークに WWW サーバーを設置して、ユーザはブラウザで社内情報を閲覧します。

◆ インタフェース
インタフェースはシンプルなメカニズムであり、ソフトウェア内の異なる部分を接続する働きを持ちます。 具体例として、API はオペレーティングシステムとともに提供され、システムレベルのサービスに対するプログラミング言語からのアクセスを実現します。なお、COM および OLE における"インタフェース"の定義は、オブジェクト間のコミュニケーションを定義するメソッドのコレクションとなります。

エ

◆ エクスポーズ
エクスポーズとはオブジェクトの機能を開示することです。このエクスポーズを利用して OLE オートメーションは実行されます。

オ

◆ オートメーション
OLE2.0で提供されるオートメーションは、複数のコンポーネントを結合し、アプリケーションを自動化するための機能です。 オートメーションはコンポーネントから他のコンポーネントの持つサービスを、ダイナミックに利用する環境を実現します。 OLE オートメーションに対応したコンポーネントは、オートメーションオブジェクトと呼ばれるアブストラクションを利用して、自分自身のサービスをエクスポーズします。

カ

◆ クラス
オブジェクトのタイプ指定に関するデータとメソッドの定義を行う、抽象的なデータタイプをクラスといいます。

◆ クラスライブラリ
オブジェクト指向型言語において、あらかじめ設計されたクラスモデルを提供するライブラリのことです。

キ

◆ キャッシュ
データへのアクセスを効率的にするため、よく使用するデータを一時的に格納するメモリ領域です。

サ

◆ サーバ
一般に、サーバという単語はクライアントアプリケーションにサービスを提供するコンピュータを示すために使われます。しかし、OLE2 の世界では OLE コンテナに対してオブジェクトを提供する、OLE2 対応アプリケーションのことをサーバといいます。例として、グラフをワープロ内にエンベッドしたとき、グラフは OLE サーバでありワープロは OLE コンテナです。OLE2 対応アプリケーションはコンテナとサーバの双方の機能を持つことができます。なお、サーバはコンテナから独立したプロセスを持ち、OCX はコンテナのプロセス内で動作します。

ス

◆ スクリプト
オペレーティングシステム(OS)やアプリケーションソフトの機能（コマンド）を使う一連の処理手順をテキスト(文字)で記述した簡易なプログラムです。

◆ スレッド（Thread）
マルチスレッド OS におけるプログラム実行の単位です。マルチスレッド OS では、プログラム（プロセス）の中で複数のスレッドを実行することができます。同じプロセスの中のスレッド同士はメモリ資源などを共有するので、スレッド間のデータ転送は比較的容易です。

タ

◆ タイプライブラリ
タイプライブラリは COM オブジェクトのクラス識別子、インタフェース識別子、メソッド名、その引数、そのデータ型、引数の並び順、戻り値のデータ型、プロパティ名とその引数の情報など、COM オブジェクトに関する情報を OLE コンパウンドドキュメントファイル（.tlb ファイル）に格納したバイナリファイルです。

ハ

◆ ハンドル
MIDL は、OLE サーバアプリケーションを含むリモートアプリケーションに接続するために、いくつかの種類のハンドルを提供します。ハンドルは、アプリケーションにネットワーク接続をするためのプロトコル情報、アドレス、その他の情報を含みカスタマイズ可能です。

◆ ハンドラ
EXE タイプの OPC サーバは、オーバヘッドにより DLL タイプのサーバよりも応答速度が低速となります。クライアントから見た性能改善の為にローカルデータキャッシュやデータアクセス最適化機能を持つハンドラをベンダは提供することができます。このモデルは、クライアントから見た場合に DLL タイプのサーバと同様となり、実際の EXE サーバとの接続はハンドラが代行します。

フ

◆ プロトコル
データやメッセージをやりとりするために必要な手順や約束事、方式を決めたものです。

◆ プロパティ
プロパティはコンポーネントオブジェクトに組み入れられるアトリビュートです。

マ

◆ マーシャリング（Marshaling）
プロセス空間、ネットワークをまたいだ通信を行うために、CPU アーキテクチャや OS によるデータ型、ビット配列順、境界の違いを吸収するために行う正規表現への変換です。クライアントの要求はプロキシのマーシャリング機能で正規表現のパケットを送信し、サーバのスタブはこのパケットをローカル表現に逆変換してオブジェクトに渡す。戻り値はサーバからクライアントに逆向きの変換が行われれます。

メ

◆ メソッド
メソッドはオブジェクトにより提供される論理的な操作であり、オブジェクト上で機能するオペレーションは、"オブジェクトのメソッド"として定義されます。 メソッドを実行するためにオブジェクトは、receiving object と特定のメソッド名で構成されるメッセージを送信します。メソッドの名前はセレクタとも呼ばれます。

レ

◆ レジストリ
Windows NT や Windows 95 において、デバイスドライバ設定からアプリケーション設定まで、コンピュータに関するあらゆる設定情報を集中管理するデータベースのことです。

◆ 列挙オブジェクト
ある項目のリストを順次処理する場合に使用されるオブジェクト。項目のリストを列挙オブジェクト（Enumerator）として作成し、このオブジェクトが持つインタフェース（Ienumxxx）を通して処理を行います。

10．2関連文献

オンライン：

OPC1.0仕様
ftp://zilker.net/pub/opc/OPCFinal.zip

日本OPC協議会ホームページ (日本OPC協議会情報)
http://www.asia.microsoft.com/japan/partners/industry/opchome.htm

OPC-F ホームページ
http://www.opcfoundation.org/

OPCサンプルコード 他 (株式会社 インテリューション)
http://www.intellution.co.jp/internet_opc_foundation/

雑誌 (OPC関連)：

プラントエンジニア 1996年 6月号
プロセス制御システムの可能性を拡げるOPC

計装 1996年 9月号
32ビット対応分散制御ソフトウェアの進展とOPCへの取り組み

オーム社エレクトロニクス 1996年11月号
OPCの目指すもの

Open Network 1996年 12月号
マイクロソフトの企業情報システム戦略

日経メカニカル別冊 ディジタルファクトリ
生産プロセスのためのOLE

オートメーション 1996年11月号
OPCと汎用監視制御用パッケージソフトFIXへの適用

オートメーション 1996年11月号
FA/PAにおけるOPCの適用

計装 1997年1月号
“OLE for Process Control「ＯＰＣ」ーその技術概要と期待される効果”

オートメーション 1997年1月号
DCSとその周辺におけるオープンソフトウェア利用システム技術

オートメーション 1997年1月号
RELAY! ESSAY 計装と情報処理 ― OPC(OLE for Process Control)

計装 1997年9月号
“OPCサポートによるバッチ制御ソフトウェアと統合設定環境”

計装 1998年8月号
“OPC技術がもたらすフィールド計装の新局面”

書籍･雑誌 (OLE/COM/ActiveX関連):

Inside OLE, second edition (1995)
OLEのバイブルというべきもの。約1200ページの大部、OLEを駆使するためには必読の書。1995年に出た本なので、DCOMやActiveXは触れられていない。

Understanding ActiveX and OLE (1996) Microsoft Press, $22.95 ISBN 1572312165
"A guide for developers & managers"と銘打っているだけあって、320ページほどの分かりやすく、最新機能(DCOM、ActiveX、など) の解説ドキュメント。日本語版も出ている。

OLE2 Programmer's Reference(1994)
プログラマー向け。日本語版も出ている。

OLE Automation Programmer's Reference(1996)
プログラマー向け。日本語版も出ている。

DCOM デベロップメントガイド (1998)
COMオブジェクトの作成からDCOMの導入
出版社:翔泳社 著者:フランク･E.レッドモンド ISBN:4881356038

DCOMガイドブック (1997)
出版社:オーム社 著者:古山一夫 ISBN:4274062163

Inside COM (1997)
出版社:アスキー 著者:デイル・ロジャーソン ISBN:4756121764

パワープログラミング MFC COM (1998)
出版社:ソフトバンク 著者:テンプルマン、ジュリアン ISBN:4797305541

OLEコントロール入門 (1998)
出版社:プレンティスホール出版 著者:プオポロ、ジョン・P ISBN:4894710382

UNDERSTANDING ActiveX AND OLE (1997)
出版社:アスキー 著者:チャペル、デビッド ISBN:4756117066

Inside OLE (1997)
出版社:アスキー 著者:クレーグ・ブロックシュミット ISBN:4756116183

OLE オートメーションプログラマーズリファレンス (1996)
出版社:アスキー 著者:Microsoft Corporation ISBN:4756110584

Microsoft System Journal December 1995
・“OLE Q&A” (日本語版 1996年2月号)

マイクロソフトシステムジャーナル日本版 No.44 August 1996
・Windows NT4.0の分散COMと新しいOLE機能
・コンポーネントソフトウェアデザインに対するOLEとCOMの解答

マイクロソフトシステムジャーナル日本版 No.45 October 1996
・ActiveXプログラミング

マイクロソフトシステムジャーナル日本版 No.47 February 1997
・IDLを使ったCOMインターフェイス定義での属性

マイクロソフトシステムジャーナル日本版 No.49 June 1997
・COMサーバの寿命を正しく管理するには

マイクロソフトシステムジャーナル日本版 No.51 October 1997
・アプリケーションのコンポーネント化と拡張
・VBやC++で書いたクライアントにオブジェクトを正しく作成するためのメカニズムを提供する方法/COMでシングルトンオブジェクトを実装するには

マイクロソフトシステムジャーナル日本版 No.52 December 1997
・オブジェクト指向ソフトウェアを単純化するCOM+ランタイムサービス Part I

マイクロソフトシステムジャーナル日本版 No.53 February 1998
・オブジェクト指向ソフトウェアを単純化するCOM+ランタイムサービス Part II

マイクロソフトシステムジャーナル日本版 No.54 April 1998
・メッソドのなかから呼び出してきたコードのホストアドレスを知る方法

マイクロソフトシステムジャーナル日本版 No.54 April 1998
・メッソドのなかから呼び出してきたコードのホストアドレスを知る方法

マイクロソフトシステムジャーナル日本版 No.55 June 1998
・ネットワークデータパケットを分析してDCOMプロトコルを理解する

Cマガジン 1996年 11月号
・ActiveX特集の中で、ActiveXの基礎技術として11ページにわたりDCOMを解説

Cマガジン 1997年 8月号
・最新ActiveXアプリケーション開発手法

Cマガジン 1997年 12月号
・特集ATL(Active Template Library) 2.1

日経コンピュータ 1996年11月25日、12月9日
・分散オブジェクト技術 DCOMの全貌(上)、(下)

日経バイト 1997年7月25日
・分散オブジェクトを検証する

日経デジタル・エンジニアリング 1998年5月
・計測制御装置をOLE/COMで結ぶ まずはデータの読み書きで実現