

SEMI E96-1101(Reapproved 0307) CIM フレームワークテクニカルアーキテクチャに関するガイド

免責事項: この SEMI スタンドダードは、投票により作成された英語版が正式なものであり、日本語版は日本の利用者各位の便宜のために作成したものです。万が一英語と日本語とに差異がある場合には英語版記載内容が優先されます。

本ガイドは、global Information & Control Committee で技術的に承認されている。現版は 2006 年 11 月 21 日、global Audits and Reviews Subcommittee にて発行が承認された。2007 年 2 月に www.semi.org で、そして 2007 年 3 月に CD-ROM で入手可能となる。初版は 2000 年 2 月発行、前版は 2001 年 11 月に発行された。

注意: 本文書は SEMI PR5-0699 を全面的に置き換えたものである。

1 目的

1.1 このガイドにはコンピュータ統合生産(CIM)環境においてアプリケーションコンポーネントが協調し、そのコンポーネントを実用的に統合する為に必要な作業を削減できるテクニカルアーキテクチャが記述されている。CIM フレームワークテクニカルアーキテクチャガイドは公開されている分散オブジェクトコンピューティング仕様が元になっている。ここではコンポーネントの相互運用性、代替性、および拡張性を向上させるのに必要なテクニカルインフラストラクチャに関する製造システムの要求事項を定義してある。本文書はコンポーネントを規定するためのガイドであり、ここには基礎となる分散オブジェクト通信インフラストラクチャの使用に関するオプションが書かれている。

1.2 本文書は SEMI コンピュータ統合生産(CIM)フレームワークスタンダードの技術的な基礎を築くためのガイドである。ここではオブジェクト指向とフレームワークテクノロジーを利用するコンポーネントベースのアーキテクチャについて記述してある。このテクノロジーを使うことによってコンポーネントの相互運用性および代替性、アプリケーションの拡張性および再利用の実現が容易になる。本文書ではフレームワークテクノロジーに必要なサポートを行う際の分散オブジェクト通信インフラストラクチャの役割を規定している。CIM フレームワーク仕様を別のインフラストラクチャテクノロジーへマッピングするための方法もこのテクニカルアーキテクチャで扱っている。しかし、そのマッピングは規定されるものではない。新たに出てきているテクノロジーへのマッピングを定義するには今後も作業が必要だろう。特定のソフトウェアに関して解決する必要がある実装上の問題は本ガイドの範囲外である。

1.3 テクニカルアーキテクチャだけに関してこのガイドに忠実であってもアプリケーションの相互運用性は得られない。テクニカルアーキテクチャは相互運用性の土台となるものだが、それには以下の制限がある:

- 複数のインフラストラクチャ・実装から選ぶことが可能で、そういった複数の環境の間での相互運用性は保証されていない。
- テクニカルアーキテクチャの範囲は故意に最も基本的なインフラストラクチャ要求事項だけに制限しており、それ以上の技術上の問題は将来のガイドの改訂版あるいは実装者の裁量に任せている。
- CIM フレームワークドメインアーキテクチャ仕様への準拠はドメインコンポーネントの相互運用性に関しても要求される。
- 別々のサプライヤが開発したコンポーネントの相互作用に一貫性を持たせるには、(挙動に関する制約および協力パターンを含む)より完全なコンポーネントの意味が必要である。

1.4 テクニカルアーキテクチャに関するガイドは CIM フレームワーク仕様の目標に到達するために必要な基礎だが十分なものではない。ガイドは発見された技術上の要求事項に特定の解決策で対応させようとはしていない。なぜならその要求事項を満足させる実装は一つだけではないからである。テクニカルアーキテクチャが行うのはむしろ、CIM ソフトウェアサプライヤと顧客の双方が考えなければならないきわめて重要な技術上の要求事項を特定することである。スタンダード案は実装が提供しなければならない技術的性能を特定しているが、実装のオプションはオープンにしている。各性能の実装を提供し説明するのはサプライヤの仕事であり、特定の実装を自分の工場でするかどうかを査定するのは顧客の仕事である

1.5 このガイドは CIM フレームワークコンポーネントのドメイン仕様をサポートし実現させるために分散コンピューティングインフラストラクチャが提供するサービスを得る為に必要な技術について指導している。その技術とは以下の通りである:

- **分散オブジェクト通信** – CIM フレームワークインタフェースをサポートしている実装が、分散されていると思われる他の実装の場所をトランスペアレントに特定し、スタンダードな CIM フレームワークオペレーションを要求するメッセージを交換するための基本的なサービスを提供する。インタフェース定義言語 (IDL) は統合および相互運用の準備が整った準拠実装に自動的に変換できる CIM フレームワークインタフェースの形式を規定する。
- **例外の宣言** – 要求されたオペレーションは予期されてはいたが異常な結果に終わったことを要求者に通知するリターンメッセージの形式および構造を識別する。
- **イベント仕様** – 予期されていた状態の変更が発生したことを CIM フレームワークオブジェクトに報告するための伝達メカニズム、識別規則、およびデータ構造を規定する。
- **分散トランザクション** – CIM フレームワークコンポーネントの境界を越えて移動する単位作業のスタート、完了、あるいはロールバックを調整するのに必要なメカニズムを定義する。
- **コンポーネントマネージャサポート** – CIM フレームワークが規定するインタフェースをサポートするオブジェクトのインスタンスを生成、配置、あるいは削除 (およびそういったオブジェクトの集まりを管理) するのに必要なコンポーネントレベルのオペレーションを識別する。

2 範囲

2.1 対象者

2.1.1 本文書は CIM フレームワーク仕様に適合するコンポーネントおよびアプリケーションの開発者、および MES システムのインテグレータを対象としている。また、CIM フレームワークアーキテクチャおよびガイドを実装の経験に基づいて進化させることに協力的なシステムアーキテクトも対象としている。テクニカルアーキテクチャ用ガイドで中心となるのは CIM フレームワークの製造ドメインコンセプトではなく、CIM フレームワークのアーキテクチャ上の目標をサポートするソフトウェアテクノロジーである。テクニカルアーキテクチャの考え方は SEMI E81 を補足している。

2.2 本文書で扱っていないアーキテクチャ上の問題

2.2.1 本文書で扱っていないアーキテクチャ上の問題は数多くある。理由はそういった問題が現在 CIM フレームワークスタンダードの範囲外であること、およびスタンダードが改訂されてもその範囲に含まれることにはならないだろうと予想されているからである。CIM フレームワークテクニカルアーキテクチャの上に積み重ねられるプロダクトアーキテクチャはそういった余分なアーキテクチャ上の問題も処理する必要があるため、ここでは問題をリストにしている。そういった問題に関しては、インフラストラクチャテクノロジーの分野から出て来つつある、より一般的な別の仕様が必要なスタンダードを提供するものと予想される。CIM フレームワークドメイン仕様では、コンポーネント仕様をサポートするのにそういった分野で特定のものに準拠することは要求していない。

2.2.2 永続性

2.2.2.1 永続性とは、オブジェクトの現在状態を如何なる場合でも保持し続ける機能で、後に初期化された際に以前の状態に復元可能な能力を指す。オブジェクトの永続性に関してはさまざまなオペレーションがあり、クロスリファレンスが設定されているオブジェクトのストラテジーおよびメカニズムがオブジェクト間で連携していなければいろいろな問題が発生する。CIM フレームワークでは永続性は実装のメカニズムとして除外している。

2.2.3 システムパフォーマンス

2.2.3.1 システムパフォーマンスはシステムの実行に関してハードウェアおよびソフトウェアのプラットフォームの選択に依存するところが多い。期待する動作環境に適したシステムパフォーマンスおよびスケーラビリティを検証するためにテストを行う必要がある。パフォーマンスチューニングメカニズムあるいは計測ツールは実装によって異なるメカニズムとして CIM フレームワーク仕様から除外されている。

2.2.4 データ複製

2.2.4.1 データ複製とは、ある状況での耐故障性を高める、あるいはシステムパフォーマンスを向上させるために使われるテクニックである。CIM フレームワークでは複製ストラテジーの仕様を実装によって異なるメカニズムとして除外している。

2.2.5 変更管理

2.2.5.1 変更管理とはシステムコンフィグレーションへの変更を取り入れコントロールする能力である。CIM フレームワークでは変更管理を文書コントロールのドメインコンテキストに含めているが、MES ソフトウェアコンフィグレーションそのものに関する変更管理の処理は除外している。

2.2.6 外部化

2.2.6.1 外部化は持続性の形式を作るあるいはつながっていない実装の間でオブジェクト状態を伝えるのに使用できる。オブジェクトが自分の持つデータおよび状態を外部化できればメモリから失われたデータおよびオブジェクトに関する状態が回復できる。CIM フレームワークでは外部化を実装によって異なるメカニズムとして除外している。

注意: このスタンダードはその使用に関連したすべての安全問題を取り扱うことを意図していない。このスタンダードの使用者は、その責任において、適切な安全および健康上実施すべき事柄を確立し、また使用前に法規制やその他の制限への適用性を判断するものである。

3 制限

3.1 CIM フレームワークは競争力を持つ活気ある産業界のニーズを満たすために進化し続ける必要がある。このフレームワークの内容は多数の商業用ソフトウェアサプライヤーおよびその顧客の現実の開発経験を大量に集めたものである。ここに記されている仕様はそういった会社のプロダクトアーキテクチャだけでなく顧客の要求事項も反映している。CIM フレームワークに基づいたプロダクトが開発されていくにつれてこの進化も進んでいかなければならない。

3.1.1 このガイドには不完全な点があり、それは将来の改訂版で対応しなければならないものである。欠けている点は下記の通りである。

3.2 別の分散コンピューティングインフラストラクチャへのマッピング

3.2.1 CIM フレームワークは MES ソフトウェアコンポーネントに関する仕様を規定している。この仕様は一般化された製造システムの要求事項に関して規定されており、さまざまなテクニカルインフラストラクチャを土台として実装される。このガイドの意図は、仕様は厳しく、インフラストラクチャの実装の選択は柔軟に、ということだがこの2つの目標は衝突することが多い。マッピングテクニックを使うとテクノロジーの境界を越えてアプリケーションを統合するという作業が一層複雑になる。

3.2.2 CORBA^{®1}あるいはMicrosoft^{®2}分散コンポーネントオブジェクトモデル(DCOM)のどちらかを使っている準拠実装はこの仕様を実装にマッピングすることでうまくいくと思われるが、クロスインフラストラクチャ統合では(例えばトランザクションモデルの融合などは)非常に難しいことがわかっている。ここに記されているマッピングにより実装はより多様になるが、選ばれたテクノロジーの全部が、均質でない一つの実装で簡単に協調するという保証はない。

3.2.3 静的な呼び出しではDCOMマッピングでOMGインタフェース定義言語(IDL)^{TM3}仕様からそのまま変換されるが、動的な呼び出しではMicrosoft OLE自動化インタフェースが必要になるかもしれない。CIMフレームワークをOLE自動化インタフェースへマッピング出来るようにするには大きな危険が伴う可能性がある。動的な呼び出しに関する要求事項はそれを念頭において評価しなければならない。

1 CORBA はアメリカ合衆国その他の国での Object Management Group, Inc. の登録商標である。

2 Microsoft はアメリカ合衆国その他の国での Microsoft Corporation, Inc. の登録商標である。

3 OMG インタフェース定義言語 (IDL) はアメリカ合衆国その他の国での Object Management Group, Inc. の商標である。

3.2.4 DCOM マッピングに関するもう一つの問題は例外である。DCOM はリターン値 HRESULT を使って例外を返す。CIM フレームワークオペレーションの多くはすでにリターン値を使っているため、オペレーション結果に関するリターンメカニズムを構成し直さない限り HRESULT を返すことはできない。

3.2.5 最後に、CIMフレームワークインタフェースはCORBA Interworking Architecture⁴を使ってマッピングできるという事を検証するための詳細な解析がまだ行われていない。特にOLE自動化へのマッピングの解決に関する解析が行われていない。

3.3 ビジネスルール

3.3.1 ファクトリオブジェクトの管理にはビジネスルールを使う必要がある。ビジネスルールとは、ある状況下でファクトリイベントに対応するために適用しなければならない共通のビジネス行動を表したプロシージャである。例えば「24 時間以内にメンテナンスが予定されているマシンにプロセスジョブを割り当ててはいけない」といったものである。ファクトリシステムの実装は普通ビジネスルールをイベント主導の ECA (イベント-コンディション-アクション) ルールあるいは ECAA (イベント-コンディション-アクション-代替アクション) ルールとして規定する。例えば：

- イベント-プロセス仕様の編集を要求する；
- コンディション-ユーザアクセス権が無効；
- アクション-アクセス拒否；
- (代替アクション-アクセスを拒否してセキュリティ違反を報告する)。

3.3.2 ビジネスルールは連続したプロセス定義のシーケンスロジックでも表現できる。この場合ビジネスルールはファクトリでの作業フローに影響を及ぼすシーケンス決定基準を定義する。

3.3.3 ビジネスルールは本ガイドの将来のバージョンで扱う予定である。

3.4 セキュリティおよびアクセスコントロール

3.4.1 ビジネスプロセスおよびプロダクト仕様に関連した機密情報の管理では MES 実装 (特に分散システム) が あるレベルのセキュリティとアクセスコントロールサービスを提供する必要がある。普通は下記のようなサービスが必要である：

- 機密情報を要求するファクトリオブジェクトを識別し、本人であることを確認する
- ID および権限に基づいて情報あるいはオペレーションへのアクセスを許可する
- セキュリティ関連の監査履歴を提供する
- 安全な通信 (中断されにくく、悪意のある修正あるいは不注意による修正が行われにくい通信) を提供する
- 企業のセキュリティポリシーを執行する。

3.4.2 セキュリティおよびアクセスコントロールは本ガイドの将来の改訂版で扱う予定である。

3.5 国際化

3.5.1 国際化に関連した問題を扱っている仕様は OMG を含むいくつかの団体から出されている。本ガイドでは国際化機能を CIM フレームワーク仕様に組み込むことが必要かどうかおよびそれが可能かどうかを扱う必要がある。

3.5.2 国際化は本ガイドの将来の改訂版で扱う予定である。

3.6 オブジェクトプロパティ

3.6.1 オブジェクトプロパティとは、プロパティが付随しているオブジェクトのインタフェースを変えずに、追加属性 (データ) をそのオブジェクトに動的に関連させるテクニックである。これは将来の CIM フレームワーク仕様で扱うと思われる便利な動的拡張メカニズムとして利用できる。

⁴ Object Management Group. The Common Object Request Broker: Architecture and Specification, Revision 2.2, Object Management Group, 492 Old Connecticut Path, Framingham, MA: Object Management Group, 1998.

3.6.2 オブジェクトプロパティは本ガイドの将来の改訂版で扱う予定である。

3.7 オブジェクト収集および検索

3.7.1 オブジェクト収集および検索によって、あるオブジェクトグループに関するデータを集めるための柔軟なアクセスが可能になる。この性能はコンポーネントマネージャインタフェースで見られる、限られた収集オペレーションに替わるものとなるかもしれない。

3.7.2 オブジェクト収集および検索は本ガイドの将来の改訂版で扱う予定である。

4 参照スタンダードおよび文書

4.1 SEMI スタンダード

SEMI E81 — Provisional Specification for CIM Framework Domain Architecture

注意：別途指示のない場合、引用されたすべての文書は最新の出版バージョンである。

5 用語

5.1 略語

5.1.1 *ACID* — 原子(最小)単位性・一貫性・独立性・堅牢性 (Atomicity Consistency Isolation Durability)

5.1.2 *CIM* — コンピュータ統合生産 (Computer Integrated Manufacturing)

5.1.3 *ECA* — イベントーコンディションーアクション (ルール) (Event-Condition-Action)

5.1.4 *ECAA* — イベントーコンディションーアクションー代替アクション (ルール) (Event-Condition-Action-Alternative Action)

5.1.5 *ENS* — イベント通知システム (Event Notification System)

5.1.6 *ERP* — 企業資源計画 (Enterprise Resource Planning)

5.1.7 *GUI* — グラフィカルユーザインタフェース (Graphical User Interface)

5.1.8 *MES* — 製造実行システム (Manufacturing Execution System)

5.1.9 *ODL* — オブジェクト定義言語 (Object Definition Language)

5.1.10 *OMA* — オブジェクト・マネジメント・アーキテクチャ (Object Management Architecture)

5.1.11 *OTS* — オブジェクトトランザクションサービス (Object Transaction Service)

5.2 定義

5.2.1 *アプリケーション(application)* — 1. エンドユーザあるいは別のアプリケーションにドメイン独自の機能を提供する相互運用可能なオブジェクトの集合でできている一つあるいはそれ以上のプログラム。2. 相互作用を行うオブジェクトの集合でできている一つあるいはそれ以上のプログラムが提供する機能。

5.2.2 *アプリケーションインタフェース(application interface)* — アプリケーションあるいはアプリケーションプログラムが提供するインタフェース。

5.2.3 *アプリケーションオブジェクト(application object)* — アプリケーションインタフェースを実装するオブジェクト。

5.2.4 *アーキテクチャ(architecture)* — あるプログラム／システムのコンポーネント、その相互関係、そのデザインおよび進化をずっと支配する原則およびガイドラインの構造。

5.2.5 *属性(attribute)* — オブジェクトと値の間にある識別可能な関係。属性は値をセットし検索するための関数を持つ場合がある。

- 5.2.6 挙動(*behavior*) — 要求されたサービスを実行することによって発生する影響およびその結果(例, オブジェクトの状態の変化)。
- 5.2.7 結合(*binding*) — プラットフォームテクノロジーとその他の実装独自の基準を独自に選択したもの。。
- 5.2.8 クラス(*class*) — いくつかのオブジェクト実装が共有している共通構造および共通挙動。
- 5.2.9 クライアント(*client*) — 別のオブジェクトにメッセージを送信するあるいは状態を参照して, そのオブジェクトのサービスを利用するオブジェクト。
- 5.2.10 集合(*collection*) — 自分以外の(複数の)オブジェクトについての記述およびそのオブジェクトを管理するためのサービスを持ち, 関連した一つのオブジェクトグループとしてその(複数の)オブジェクトへのアクセスを提供するオブジェクト。
- 5.2.11 コンポーネント(*component*) — カプセル化されたオブジェクトおよび/あるいは別のコンポーネントを集めた再利用可能なパッケージで, 明確に規定された公開インタフェースを持つ。コンポーネントは標準化およびCIMフレームワークとの代替性の要因の一つである。
- 5.2.12 コンピュータ統合生産(*Computer Integrated Manufacturing*) — コンピュータの情報処理能力を高めて製造情報を管理し製造オペレーションの実行をサポートあるいは自動化しようというアプローチ。
- 5.2.13 準拠(*conformance*) — プロダクト, プロセス, あるいはサービスの実装でスタンダードあるいは仕様に適合していること。
- 5.2.14 準拠要求(*conformance requirement*) — ある実装がその仕様に準拠するために要求する挙動および/あるいは性能の規定を特定すること。
- 5.2.15 準拠実装(*conforming implementation*) — 規定されている関連準拠要求をすべて満たす実装。
- 5.2.16 イベント(*event*) — 重要な出来事の発生を意味する非同期的メッセージ。例えば状態変化あるいは新規オブジェクト生成。
- 5.2.17 イベントチャネル(*event channel*) — 発行されたイベントをそのイベントに関心を持つオブジェクトへ転送する中間オブジェクト。
- 5.2.18 例外(*exception*) — オペレーションをコールしたクライアントに, そのオペレーションを実行する際に異常な状況が発生したことを通知するのに使われるインフラストラクチャメカニズム。
- 5.2.19 拡張性(*extensibility*) — アーキテクチャの完全性およびコンポーネントのスタンダードへの準拠は保持したまま, 既存のコンポーネントを拡張あるいは特化し, 新しいオブジェクトクラスあるいはコンポーネントを追加できること。
- 5.2.20 フレームワーク(*framework*) — 特定のドメインで相互運用的なサービスおよび機能を提供するクラスあるいはコンポーネントの集合。
- 5.2.21 実装(*implementation*) — クラス, オブジェクト, あるいはモジュールを内側から見たもので非公開挙動を含む。インタフェースを実現する特定のコードおよび機能。
- 5.2.22 実装準拠ステートメント(*implementation conformance statement*) — 実装あるいはシステムのサプライヤが出すステートメントで, 一つあるいはそれ以上の仕様に準拠していることを主張し, どの性能が実装されているかを述べたもの。特に関連したオプション性能および限界を含む。
- 5.2.23 インフラストラクチャ(*infrastructure*) — 分散オブジェクトのライフサイクルや相互の協調をサポートするサービス, 設備, および通信メカニズム。
- 5.2.24 インヘリタンス(継承)(*inheritance*) — 既存のクラス, タイプ, あるいはインタフェースから新しいクラス, タイプ, あるいはインタフェースを派生させる性能。例えば, 派生したクラス(「サブクラス」)は土台となるクラス(「スーパークラス」)のインスタンス変数およびメソッドを継承し, 新しいインスタンス変数およびメソッドが追加される場合もある。CIMフレームワークではインヘリタンスはクラスの実装ではなくインタフェースおよびオペレーションに関するインタフェース仕様に適用される。

5.2.25 インスタンス(*instance*) — 状態、挙動、およびアイデンティティを持つソフトウェアエンティティ。インスタンスおよびオブジェクトという用語は交換可能である。あるインタフェースが規定したオペレーション、シグニチャ、意味をオブジェクトが提供するなら、そのオブジェクトはそのインタフェースのインスタンスである。ある実装がオブジェクトの挙動を提供するなら、そのオブジェクトはその実装のインスタンスである。

5.2.26 インタフェース(*interface*) — クラス、オブジェクト、あるいはモジュールを外部から見たもの。自分の抽象性は強調するが構造および内部挙動は隠す。インタフェース定義に属性およびオペレーションの意味が含まれるのが理想である。

5.2.27 相互運用性(*interoperability*) — 2つのアプリケーションあるいは一つのアプリケーションのいくつかの部分が協力できること。CIM フレームワークでは、アプリケーションコンポーネントは規定された関係をサポートできる、データを共有できる、互いの挙動(オペレーション)を呼び起こせる、例外を返せる、およびイベントを交換できることが要求されている。

5.2.28 ライフサイクル(*lifecycle*) — 生成、削除、複製など、オブジェクトの一生。

5.2.29 メッセージ(*message*) — オブジェクト指向のシステムでは、メッセージは、クライアントオブジェクトがサーバーオブジェクトのオペレーションで規定される挙動を引き起こす手段である。

5.2.30 メッセージバス(*message bus*) — コンポーネント実装の中のオブジェクト同士の分散通信を提供するソフトウェアインフラストラクチャ。オブジェクトリクエストブローカ(Object Request Broker)⁵、Microsoft DCOM、JavaRMI (Remote Method Invocation)、あるいはオブジェクト間のメッセージ伝達用のインフラストラクチャ参照。

5.2.31 名前-値のペア(*name-value pair*) — 名前を任意の値に関連させるデータ構造で、通常は名前を基にした検索によって情報を伝えるための拡張メカニズムとして使われる。

5.2.32 ネームスペース(*namespace*) — ネームスペースとは限られた数の名前の集合で、その集合の中ではそれぞれの名前は絶対にユニークであるという制約がある。

5.2.33 オブジェクト(*object*) — 状態、挙動、およびアイデンティティを持つソフトウェアエンティティ。インスタンスおよびオブジェクトという用語は交換可能である。あるインタフェースが規定したオペレーション、シグニチャ、意味をオブジェクトが提供するなら、そのオブジェクトはそのインタフェースのインスタンスである。ある実装がオブジェクトの挙動を提供するなら、そのオブジェクトはその実装のインスタンスである。

5.2.34 オブジェクトサービス(*object services*) — 分散オブジェクトに基づいたプログラムで使われることが多い一般的なサービスのためのインタフェース。

5.2.35 オペレーション(*operation*) — オペレーションとは要求可能なサービスをオペレーション識別子によって識別される仕様実体である。シグニチャは要求すべきオペレーションに対してパラメータの妥当性と戻り結果の妥当性を記したものである。それには例外も含まれる。

5.2.36 永続オブジェクト(*persistent object*) — 自分を生成したプロセスあるいはスレッドが消滅した後も存続できるオブジェクト。

5.2.37 プロダクティブエンティティ(*productive entity*) — プロダクティブエンティティとは、物理的ユニットの抽象概念で、生産プロセスに何らかの形で含まれている(たとえば、生産または支援装置)。プロダクティブエンティティはそれ自身内部論理を持ち、この論理にアクセスするソフトウェアインタフェースを提供する。

5.2.38 クエリー(*query*) — サーバーからの何らかの情報(プロダクティブエンティティの状態)に興味を示すクライアントがサーバー(たとえば、プロダクティブエンティティ)に送る情報。クエリーはアーギュメントを持つことも、持たないこともあるが、常に答えを持っている。クエリーの意味するところは、サーバーから情報は返されるが、クエリーはそのサーバー状態を変える影響力を持たないことである。

5.2.39 サービス(*service*) — プロバイダが規定した動作中に実行する、サービスプロバイダが与える機能

5.2.40 サービスプロバイダ、サーバー(*service provider, server*) — 公表された動作が規定するように、他のオブジェクトにサービスを与えるオブジェクト

5 Object Request Broker および ORB はアメリカ合衆国その他の国での Object Management Group, Inc. の登録商標である。

5.2.41 シグニチャ(signature) — シグニチャとはあるオペレーションの名前、パラメータ、リターン値、および例外である。

5.2.42 代替性(substitutability) — あるサプライヤからのコンポーネントと別のサプライヤからの機能的に同等なコンポーネントを、システムの別のコンポーネントあるいは入れ替えるコンポーネントのクライアントに影響を与えずに、入れ替えられること。

5.2.43 トレーダサービス(trader service) — 機能に関連したプロパティをそれぞれ持っている名前の集合、およびその集合を操作および検査するためのメソッドの集合。

5.2.44 タイプ(type) — オブジェクトの集合に共通したプロパティおよび挙動を記述したもの。タイプは共通インタフェースでオブジェクトを分類したもの。クラスは共通実装でオブジェクトを分類したもの。

6 テクニカルアーキテクチャの指針

6.1 コンピューティング基盤技術が CIM フレームワークのアプリケーションに分散コンピューティング環境を提供する。この基盤技術にはオペレーティングシステム、ネットワーキングと通信、データ保持とアクセス、ユーザインタフェースおよびプレゼンテーションサービス、イベントの伝達、システム管理、およびその他多くの要素が含まれる。こういった多くの基盤技術の中で、CIM フレームワークの実装の内、適合とされるものの間で統合を促し、しかも能率的なものにするため、スタンダード化するのに必要なキイとなるサービスの、小さなサブセットをテクニカルアーキテクチャの指針は規定する。これは準拠している CIM フレームワーク実装同士のシステム統合を促進し能率化するためである。

6.2 CIM フレームワークは、仕様が公開される限り、開放された仕様が基盤技術の能力を定義するものと考えている。分散オブジェクトサービスに関して定義される開放された仕様の最大の出所はオブジェクト・マネジメント・アーキテクチャ(Object Management Architecture)である。本指針を参照するからと言って、CIM フレームワーク準拠の各製品が指針中で参照しているサービスを実装しなければならないということにはならない。こういった基盤技術を実現する実装方式は本ガイドの範囲外である。実装者が選択した実現方式は最大限の拡張性を持てる様、CIM フレームワークに対する透過性を持つべきである。

6.3 分散オブジェクト間通信

6.3.1 CIM フレームワークの文書では実装の際に装備する分散オブジェクト間通信を提供するソフトウェア基盤技術の使用を前提としている。ORB という略語は OMG の分散オブジェクト通信基盤技術記述するために元々 OMG が使ったものだが、もっと一般的な意味で使われることがある。本文書では OMG が規定したテクノロジーを指す場合のみ ORB を使用し、オブジェクト間通信のための多用な種類の分散通信の仕掛けには、より一般的な用語「メッセージバス」を使用する。メッセージバスは、オブジェクトが別のオブジェクトに要求を出し、そこから応答を受け取る様にするため使用される。オブジェクトはメッセージバスを介して(プラットフォーム)内部のあるいは別(のプラットフォーム)のオブジェクトと通信することができる。実装位置の透過性によりオブジェクトは自分が通信するオブジェクトの実装場所を頓着する必要がない。

6.3.2 メッセージバス実装の第一の基準は CIM フレームワークのコンポーネントで、(複数の)インタフェースによって規定されるメッセージを全部配信できることである。この実現には、メッセージバスは定められるインタフェースを受け入れられる、あるいはマップできる可能性を提供するのが望まれる。その中には、継承される機能、データの型、操作、オブジェクトの参照、実行時における伝送フォーマット、および伝送フォーマットからの整形(マーシャリング)の例外処置も含まれる。

6.3.3 CIM フレームワークでは、CIM フレームワークのインタフェースに関する OMG の IDL を特定のメッセージバスの実装にマッピングすることで代替メッセージバスの実装技術が補償される。The Common Object Request Broker: Architecture and Specification には CORBA から COM へ、CORBA から OLE オートメーションへのマッピングを定義した複数の章がある。それらの章は“Interworking Architecture”と呼ばれ、OMG の IDL、タイプ、および例外を COM と OLE オートメーションにある互換インタフェースへマッピングする際の詳細なルールを取り扱っている。

6.3.4 高いレベルでは DCOM メッセージバスと CORBA メッセージバスは類似のものに見える。DCOM の能力は ORB のものと大雑把には等しい。しかし低いレベルで分析すると、データの型、継承、オブジェクトの性質、お

よび例外処理において違いが出てくる。CORBA Interworking Architecture (15 章 “Interworking Architecture,” 16 章 “Mapping COM to CORBA,” および 17 章 “Mapping OLE Automation to CORBA”)では下記の取り組み方によるマッピングが取り扱われている:

- インタフェースマッピング
- インタフェース複合マッピング
- 本質性マッピング

6.3.5 CIM フレームワークインタフェースの規定に使われる OMG IDL と実装に使われるメッセージバスとの間にマッピングを行うには上記の分野も取り上げる必要がある。マッピングが特殊なものでない(即ち、複数の方法でマッピングできる)のなら、2 種類の実装が同じメッセージバスの方式を使っていいたとしても、必ずしも通信が可能でなくてもよい。CORBA Interworking Architecture は Microsoft の方式に特化したものではあるが、将来の仕掛けの間のマッピングの礎になり得よう。

6.3.6 現在の CIM フレームワークインタフェースは OMG IDL で定義されている。インタフェースは直接コンパイル可能で市販されているあらゆる ORB 製品で利用可能である。

6.3.7 Microsoft の DCOM を使った、上記の様に直接的に記述する唯一の方法は DCOM の MIDL および/あるいは OLE オートメーション ODL で CIM フレームワークのインタフェースを追加規定することだろう。この方法を採用すれば DCOM に依るメッセージバスの機能を直接サポートできることになる。例えば、パラメータ受け渡しに関する CIM フレームワークのメモリ管理上の要件“in”(6.3.15 項参照)は MIDL と DCOM で直接サポートされるだろうが、CORBA の例外モデルは DCOM のものよりはるかに内容が充実しているため、CORBA の例外事象を COM にマッピングするには DCOM のためにプロトコルを追加定義する必要が出てしまう。

6.3.8 CORBA Interworking Architecture は OMG の IDL で定義されている現在の CIM フレームワークのインタフェースから DCOM の MIDL あるいは OLE オートメーションの ODL へのマッピングを支援する。マッピングが常に同じ MIDL/ODL 展開になる様に、それに依るマッピングは詳細に記述される。現在のところ DCOM での実装と CORBA での実装との間の乗り入れ動作は視野に入っていないが(3.2 項参照)、インタフェースのマッピングという点ではやはり同じである。これは最近出てきた CORBA/DCOM 間のブリッジ製品を使った本当の相互運用性を達成するための一つのステップにも相当する。

6.3.9 Microsoft は OLE オートメーションでの拡張も提供している。このインタフェースはオブジェクト定義言語 (Object Definition Language) (ODL) で記述される。OLE のインタフェースはコンパイル時にインタフェースを知らなくても、クライアントが動的に呼び出すことが可能である。OLE のデータ型は DCOM でサポートされている型のサブセットであり、ユーザ定義に依る型はサポートしていない。マッピングの結果は DCOM と OLE オートメーションでは異なっている。OMG の IDL から OLE オートメーションへのマッピングは DCOM へのマッピングほどきれいではない。そのためインタフェースによっては OLE オートメーション実装への完璧な書き換えはできない可能性がある。従って OLE オートメーションを実装とするコンポーネントの開発者は、(OLE)オートメーションの持つ制約のために十分にはサポートされないインタフェースへの影響を説明する必要がある。

6.3.10 CORBA Interworking Architecture では CIM フレームワークで考慮すべき主な分野についてのマッピング問題を扱っている。重要な分野と言えはまずインタフェースマッピング、インタフェースコンポジションマッピング、アイデンティティマッピングおよび例外マッピングである。CORBA Interworking Architecture には上記の分野でのマッピングが一つずつ詳細に記されており、DCOM と OLE オートメーションのマッピングを分けて扱っている。こういったマッピングの問題を下記に要約する。

6.3.11 インタフェースマッピング

6.3.11.1 OMG IDL の基本型、派生データ型、およびオブジェクト参照子はかなり厳密に DCOM にマッピングされる。継承される CORBA インタフェースは複数の DCOM インタフェースとして表されることがある。CORBA での属性は DCOM インタフェースでは get 操作および set 操作へマッピングできる。

6.3.11.2 OMG IDL の基本型は、特殊な場合を除いて OLE の基本型へマッピングされる。OLE インタフェースは派生データ型をサポートしていないので、特別に用意されたインタフェースへマッピングしなければならない。CORBA オブジェクト・リファレンスは OLE オートメーションのインタフェース・ポインタへマッピングされている。