GWD-R (draft-ggf-ogsi-gridservice-33)
Open Grid Services Infrastructure (OGSI)
http://www.ggf.org/ogsi-wg

Editors:
S. Tuecke, ANL
K. Czajkowski, USC/ISI
I. Foster, ANL
J. Frey, IBM
S. Graham, IBM
C. Kesselman, USC/ISI
T. Maquire, IBM
T. Sandholm, ANL
D. Snelling, Fujitsu Labs
P. Vanderbilt, NASA

June 27, 2003

Open Grid Services Infrastructure (OGSI)

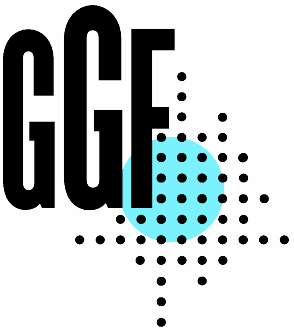
Version 1.0

この覚書の位置づけ

この文書は Open Grid Services Infrastructure (OGSI) 仕様について述べたものである。この文書の配布制限はない。

概要

Open Grid Services Infrastructure (OGSI) とは、グリッド・ウェブサービス技術に基づき、**グリッドサービス**とよばれるエンティティ上の情報の生成、管理、交換の機構を定義するものである。簡潔にいうと、グリッドサービスとはクライアントとグリッドサービスとの相互作用を定義する規約（インターフェイスと挙動）に従うウェブサービスと言える。OGSI のこれらの規約とグリッドサービスの生成、発見に関する他の機構は、先進的な分散アプリケーションにおいて通常必要とされる機能を実現する。すなわち、分散かつしばしば長期間に渡って存在する状態の制御され、故障に強く、かつ安全な管理機能を提供する。別の文書で我々は OGSI の基底となる動機、要求、構造、アプリケーションについて詳細を述べた。本文書では、技術的詳細に主眼を置き、グリッドサービスを定義する挙動と Web Service Definition Language (WSDL) インターフェイスの完全な仕様を示す。



GLOBAL GRID FORUM

office@gridforum.org
www.ggf.org

Full Copyright Notice

Intellectual Property Statement

The GGF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the GGF Secretariat.

The GGF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this recommendation. Please address the information to the GGF Executive Director (see contact information at GGF Web site).

目次

1 はじめに

Open Grid Services Architecture (OGSA) [Grid Physiology] とは、*Open Grid Services Infrastructure (OGSI)* に基づいた分散システムフレームワークを構築するために、主要なグリッド技術 (Globus ツールキット [Globus Overview] を含む) とウェブサービス機構 [Web Services Book] を統一するものである。**グリッドサービスインスタンス**とは、生存期間管理、特徴の発見、通知などを目的とした規約に従う (潜在的に一時的な) サービスである。ここで、この規約は Web Service Definition Language (WSDL) のインターフェイス、拡張、挙動として表現される。グリッドサービスは、洗練された分散アプリケーションにおける通常必要とされる機能、すなわち分散かつしばしば長期間に渡って存在する状態の制御された管理を実現する。OGSI はまたグリッドサービスを生成、発見するためのファクトリーと登録の標準インターフェイスを導入する。

OGSI バージョン 1.0 は、以下の概念を統合するために WSDL と XLM スキーマ定義を拡張するコンポーネントモデルを定義する。

- 状態を有する Web サービス
- Web サービスインターフェイスの拡張
- 状態変化の非同期通知
- サービスインスタンスへの参照
- サービスインスタンスの集まり
- XML スキーマ定義の制約能力を拡張する、サービス状態データ

本仕様において、我々は OGSA を構成するサービスの定義をサポートするために必要な、最低限かつ統合された拡張とインターフェイスを定義する。

本仕様はこの文書で完結したものであり、さらに特に Web サービスと XML は動的かつ現在もなお発展し続けている環境である。我々は将来の OGSI はより広範囲の標準に準拠することを保証する。例えば serviceData (§6) のような我々が定義する概念の多くは、XML 文書、メッセージ、ウェブーサービスにおいて見られるより一般的な概念の特別な場合として考えられる。さらに、J2EE のようなさまざまなホスティング環境において OGSI Web サービスコンポーネントモデルを実装する試みによって、現在の OGSI V1.0 仕様に対して今後変更が必要が生じるだろう。

本文書において、我々はクライアントがグリッドサービスインスタンスを生成、発見、かつ相互作用する方法を定める規約の詳細な仕様を提案する[1]。すなわち、(1) グリッドサービスインスタンスの名前付けと参照方法、(2) すべてのグリッドサービスが実装する基底共通インターフェイス（と関連した挙動）、(3) ファクトリとサービスグループに関連した追加（任意）インターフェイスと挙動、を述べる。本文書では、ある特定のホスティング環境におけるグリッドサービスの生成、管理、破壊の方法を述べること

[1]我々はクライアントという言葉をグリッドサービスを呼び出すエンティティという意味で非厳密的に用いる。多くの状況で、グリッドサービスは他のグリッドサービスのクライアントである。

はない。従って、本仕様に準拠するサービスは必ずしもホスティング環境独立ではないが、本文書で定義される規約に従う任意のクライアントプログラムは、本仕様に準拠した任意のグリッドサービスインスタンスを呼び出し可能である（当然、ポリシーと互換性を有したプロトコルバインディングによる)。

本文書の記述は簡潔なものであり、動機、要求、アーキテクチャ、グリッド・Web サービス技術との関係、他の関連した仕事、アプリケーションに関する議論については[Grid Physiology]を参照せよ。

本文書は主に以下の 4 つの項目からなる。

1. 第 1 節から第 3 節は本文書の導入であり、[Grid Physiology] を拡充するものである。
2. 第 4 節から第 7 節では、グリッドサービス仕様が Web Services Description Language を基にどのように構築されるかを紹介する。我々は WSDL 1.2 の策定が完了するまでの一時的な手段として、WSDL 1.1 に対する gwsdl 拡張を定義する。また、サービスインスタンスの状態を表現する serviceData という概念を定義し、他の中心となるグリッドサービスの概念を定義する。
3. 第 8 節から第 13 節で、GridService, HandleResolver, Notification, Factory, serviceGroup などの必須 portType と任意 portType を定義する。
4. 第 14 節から第 19 節で結論を述べる。

2 表記規則

本仕様書では、”MUST”, “MUST NOT,” “REQUIRED,” “SHALL,” “SHALL NOT,” “SHOULD,” “SHOULD NOT,” “RECOMMENDED,” “MAY,” “OPTIONAL” をキーワードとして用いる。それらの意味は RFC-2119 [RFC 2119] で定義される通りである。

本仕様は、表 1 にある名前空間接頭辞を用いる。名前空間接頭辞の選択は任意であり、意味的な違いはない。

表 1: 本仕様で用いる接頭辞と名前空間

接頭辞	名前空間
ogsi	"http://www.gridforum.org/namespaces/2003/03/OGSI"
gwsdl	"http://www.gridforum.org/namespaces/2003/03/gridWSDLExtensions"
sd	"http://www.gridforum.org/namespaces/2003/03/serviceData"
wsdl	“http://schemas.xmlsoap.org/wsdl/”
http	"http://www.w3.org/2002/06/wsdl/http"
xsd	"http://www.w3.org/2001/XMLSchema"
xsi	"http://www.w3.org/2001/XMLSchema-instance"

”http://example.org/...”や”http://example.com/...” のような一般的な形式の名前空間の名前はアプリケーションやコンテキスト依存の URI [RFC 2396]を表す。

以下の省略語と用語を本文書で用いる。

- *GSH*: 7.5 節で定義されるグリッドサービスハンドル
- *GSR*: 7.5 節で定義されるグリッドサービス参照
- *SDE*: 7.2 節で定義されるサービスデータ要素
- [Grid Physiology]で定義される **Web サービス、XML、SOAP、WSDL**

本文書では、**ホスティング環境**という用語を 1 つ又は複数のグリッドサービス実装が動作するサーバーという意味で用いる。そのようなサーバーは典型的には言語またはプラットフォーム特有のものである。例として、Unix や Windows のプロセス、J2EE アプリケーションサーバー、Microsoft .NET などがあげられる。

3 背景

[Grid Physiology]では Open Grid Services Architecture (OGSA)の全般的な動機が記されているが、本文書ではそのアーキテクチャについてより詳細に記述する。我々は OGSA の基盤を *Open Grid Services Infrastructure (OGSI)*と呼ぶ。本節では、本文書の残りの理解を助ける詳細な事柄について検討する。特に、OGSI と分散オブジェクトシステムとの関係や、さらに OGSI と現行の（発展中の）Web サービスフレームワークとの関係について議論する。また、グリッドサービスに対するクライアントサイドプログラミングパターンと、概念的なホスティング環境について検討する。

本節で説明されるパターンは OGSI によって実現可能なものであるが、**要求されるものではない**。この文書の後半で説明する詳細な事柄の理解を助けるために、これらのパターンを本節で議論する。

3.1 分散オブジェクトシステムとの関係

詳細については後述するように、グリッドサービス実装とは、アドレス可能で潜在的に状態を有するインスタンスであり、WSDL の portType によって記述される 1 つ以上のインターフェイスを実装する。グリッドサービスファクトリ（§12)は、ある portType の集合を実装するインスタンスの作成に使用可能である。個々のグリッドサービスインスタンスは、分散グリッド上の他のインスタンスとの識別を可能にする概念を持つ(§7.5.2.1)。各インスタンスは、型特有の操作によって公開される挙動と状態として特徴付けられる。このアーキテクチャはまた内省をサポートしている。すなわち、クライアントアプリケーションがグリッドサービスインスタンスに対して、そのインスタンス自身を説明する情報 （例えば、インスタンスが実装する portType の集まりなど）を返すよう求めることが可能である。

グリッドサービスインスタンスは、(潜在的に遠隔の）クライアントアプリケーションから、グリッドサービスハンドル（§7.5.2）やグリッドサービス参照（§7.5.1）を用いることによって、アクセス可能になる。これらは基本的に、(潜在的に遠隔の）実行環境に配置されている特定のグリッドサービスインスタンスへの、ネットワークワイドなポインタである。クライアントアプリケーションは、グリッドサービス参照を、グリッドサービス参照によって識別される、特定のネットワーク接続されたサービスエンドポイントに存在する特定のサービスインスタンスへ、(目的のサービス記述の portType で定義される操作で表現される）要求を直接送信することに使える。

我々は多くの状況において、クライアントスタブや補助クラスが、グリッドサービス参照を用いる際の末梢的な事柄からアプリケーションプログラマを分離することを期待する。クライアントサイドの基盤ソフトウェアには、ある操作を GSR が識別するある特定のインスタンスへ向ける責任を持つものもある。

上で紹介した特長（状態を有するインスタンス、型付きインターフェイス、グローバルネームなど）はそれぞれ、**分散オブジェクトベースシステム**の基本的特徴としてしばしば引き合いに出される。だが、OGSI では明確に要求、規定**されない**、分散オブジェクトモデルにおける（伝統的に定義されてきた）様々な他の側面も存在する。このため、我々の仕事を説明する際には、分散オブジェクトモデルや分散オブジェクトシステムといった用語を用いず、代わりに Open Grid Services Infrastructure という用語を用いる。我々は、このように我々が確立する Web サービスとグリッド技術との結合を強調する。

オブジェクト関連の事項において、OGSI は、実装の継承、サービスインスタンスモビリティ、開発アプローチ、ホスティング技術を対象外とする。グリッドサービス仕様は、インターフェイスレベルや実装レベルでの継承をサポートするオブジェクト技術を基にした実装を要求も妨げもしない。このアーキテクチャにおいて、その使用契約のクライアントサイドやサービス提供サイドにおいて、実装の継承という概念を外部に提供するという要求はない。加えて、グリッドサービス仕様はグリッドサービスインスタンスのための特定の開発法やホスティング技術を定めたり、指示したり、妨げたりしない。グリッドサービス提供者は、自身が選択した技術やホスティングアーキテクチャにおいて、サービス記述の意味的な取り決めを自由に実装可能である。我々は、J2EE、.NET、伝統的な商用トランザクション管理サーバ、伝統的な手続き型 Unix サーバなどでの実装を考えている。また、広範囲のオブジェクト指向・非オブジェクト指向プログラム言語でのサービス実装も予想される。

3.2 クライアントサイドプログラミングパターン

特に Web サービスに精通していない読者のために、我々が説明する必要があると考えるもう一つの重要な事柄は、どのように OGSI インターフェイスが呼び出されるかということである。OGSI は Web サービスフレームワークの重要なコンポーネントを活用する。すなわち、ある Web サービスに関して、多重プロトコルバインディング、エンコーディングスタイル、メッセージングスタイル（RPC vs. ドキュメント志向）、などを WSDL で用いて記述することである。 *Web Services Invocation Framework* [WSIF] や *Java API for XML RPC* [JAX-RPC]はこの機能を提供する多くの基盤ソフトウェアの例である。

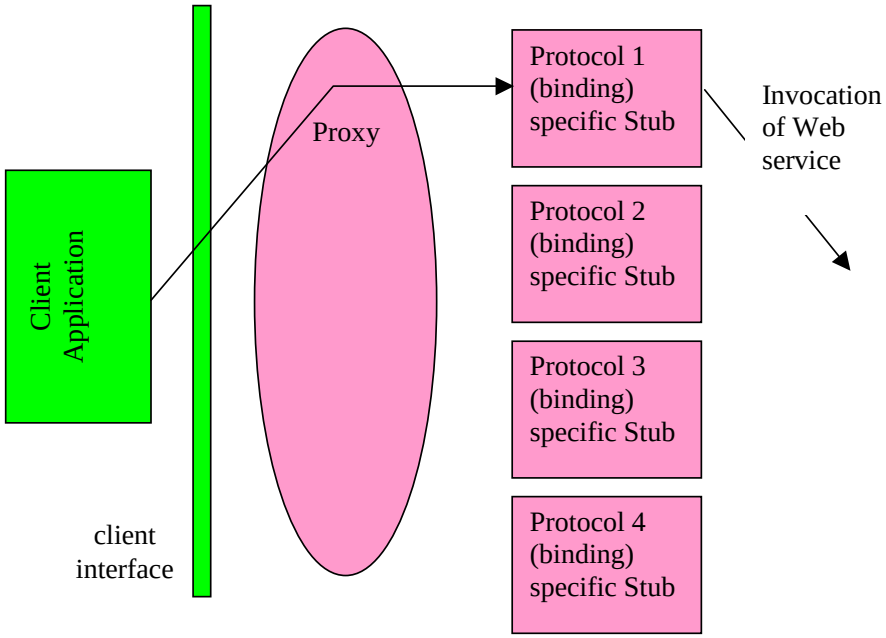


図 1:可能なクライアントサイド実行時アーキテクチャ

図 1 は OGSI の可能な(要求はされない)クライアントサイドアーキテクチャを表す。このアプローチにおいて、クライアントアプリケーションと、Web サービス(プロキシ)のクライアントサイド表現との間に明確な分離が存在する。ここで、Web サービスには、ある選択されたバインディング越しの Web サービスの呼び出しをマーシャリングするコンポーネントが含まれる。特に、クライアントアプリケーションは、クライアントサイドインターフェイスという高レベル抽象化により、Web サービス呼び出しの詳細から切り離される。

様々なツールは、Web サービスの WSDL 記述を扱うことができ、広範囲のプログラミング言語固有の構文でインターフェイス定義を生成可能である(例えば Java インターフェイスや C#)。このインターフェイスは、WSDL で提供される様々なバインディングオプションを取り込み可能な、特定の引数マーシャリングやメッセージルーティングへのフロントエンドである。さらにこのアプローチでは効率化が可能である。例えば、同じネットワークホスト上に存在するクライアントや Web サービスを検知し、ネットワークプロトコルを用いた呼び出しの準備や実行のオーバヘッドを回避可能である。

クライアントアプリケーション実行時において、**プロキシ**は遠隔サービスインスタンスのインターフェイスのクライアントサイド表現 (client-side representation) を提供する。固有のエンコーディングやネットワークプロトコル(Web サービスの用語で**バインディング**)に固有のプロキシの挙動は、**プロトコル(バインディング)固有スタブ**にカプセル化される。正しい書式の設定や認証機構のような、グリッドサービスインスタンスへのバインディング固有のアクセスに関する詳細は、このスタブで処理される。従って、アプリケーションはそれらを自身で扱うことを要求されない。

開発者が、クライアントアプリケーションをある特定のグリッドサービスインスタンスの固定されたバインディングと、直接結びつける改造コードを作成することは可能ではあるが、推奨はしないことを注意する。ある環境ではこの類の改造により得られる潜在

的な効率に対する要求があるが、このアプローチはシステムの柔軟性を大幅に失わせるため、特別な状況下でのみ用いられるべきである。

我々は、我々が説明するスタブやクライアントサイド基盤モデルが、クライアントをグリッドサービスにアクセス可能にする共通のアプローチとなることを期待する。これはアプリケーション固有のサービスと OGSA が定義する共通基盤サービスとの両方を含む。従って、グリッドサービスを利用する大抵の開発者にとって、その基盤やアプリケーションレベルのサービスは、クラスライブラリやプログラミング言語のインターフェイスといった呼び出し側にとって自然な形で表される。

WSDL と GWSDL 拡張は、異種のツールや基盤ソフトウェアを可能にすることをサポートする。

3.3 グリッドサービスハンドルと参照のクライアントによる使用

クライアントは、グリッドサービスハンドルとグリッドサービス参照を用いて、グリッドサービスインスタンスへのアクセスを達成する。グリッドサービスハンドル(GSH)は特定のグリッドサービスインスタンスを指す恒久的なネットワークポインタとみなすことができる。GSH はクライアントがサービスインスタンスにアクセスするために十分な情報を提供しない。クライアントは GSH をグリッドサービス参照(GSR)に「解決」する必要がある。GSR はサービスインスタンスにアクセスするのに必要な情報の全てを含む。GSR は、様々な理由により無効にされることがあるため、グリッドサービスインスタンスへの「恒久的な」ネットワークポインタではない。例えば、参照先のグリッドサービスインスタンスが他のサーバに移されることである。

OGSI では、クライアントによるグリッドサービスハンドルからグリッドサービス参照への解決をサポートするために、HandleResolver(§･を参照)と呼ばれる機構を提供する。図 2 は GSH から GSR へ解決を必要とするクライアントアプリケーションを示している。

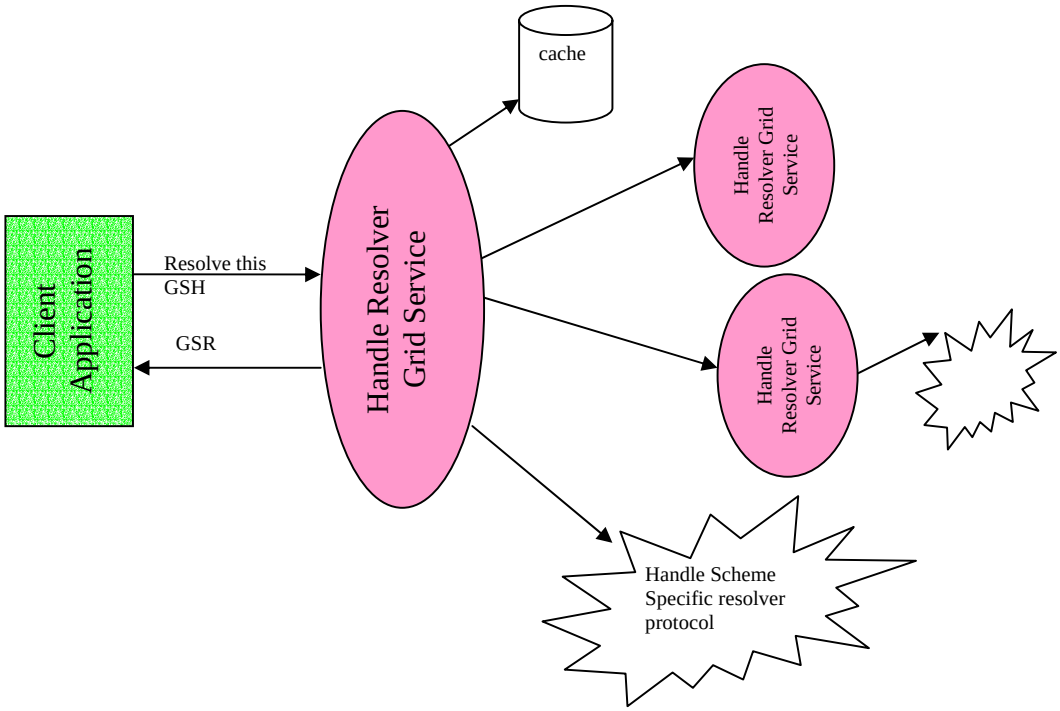


図 2: GSH の解決

クライアントは、ある規定外の機構によって特定される HandleResolver グリッドサービスインスタンスを呼び出すことで、GSH を GSR へ解決する。HandleResolver は解決を実現するために様々な手段を取れる。そのいくつかを図 2 に示す。HandleResolver は GSR をローカルキャッシュに保存しているかもしれない。HandleResolver は GSH を解決するために、他の HandleResolver を呼び出す必要があるかもしれない。
HandleResolver は、GSR へ解決する GSH の種類によって指定される、ハンドル解決プロトコルを用いるかもしれない。HandleResolver プロトコルは、解決される GSH の種類に固有である。例えば、ある種類のハンドルは GSR に解決するために、その GSH に符号化された URL に対して HTTP GET を用いることを示すかもしれない。

3.4 ホスティング環境との関係

OGSI は、特定のサービスプロバイダ側の実装アーキテクチャを指示しない。グリッドサービスインスタンスを直接オペレーティングシステムのプロセスとして実装する方法から、J2EE のような洗練されたサーバサイドコンポーネントモデルまで、多くのアプローチが可能である。前者の場合では、グリッドサービスの挙動(呼び出し、生存期間管理、登録など)に対するサポートのほとんど、もしくは全てがユーザプロセスにカプセル化される。例えば、標準ライブラリとリンクすることによってそのサポートがユーザプロセスにカプセル化される。後者のケースでは、多くの挙動はホスティング環境によってサポートされる。

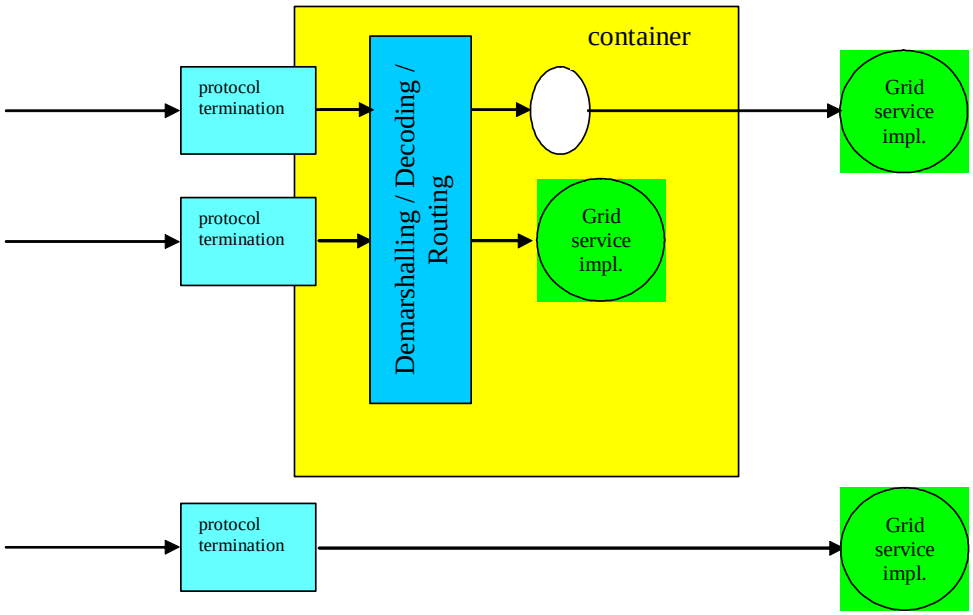


図 3: グリッドサービスホスティング環境における、引数デマーシャリング機能への 2 つのアプローチ

図 3 はこれらの相違を、引数デマーシャリング機能の実装に対する 2 つの異なるアプローチを示すことで説明する。我々は、多くのグリッドサービスで言えるように、呼び出しメッセージが、ネットワークプロトコル終端点(例えば HTTP サーブレットエンジン)で受け取られることを仮定する。この終端点は、そのメッセージ中のデータをホスティング環境で使用できる形に変換する。図 3 上部に、コンテナ管理のコンポーネント(例えば J2EE コンテナの EJB)と関連付けられた、2 つのグリッドサービスインスタンス(楕円)を示す。ここで、到着したメッセージはこれらのコンポーネントに送られる。この際、コンテナは、ある形式（XML/SOAP メッセージなど）からネイティブプログラミング言語によるコンポーネントの呼び出しへ、デマーシャリングやデコーディングする機能を提供する。ある状況では(下部の楕円)、グリッドサービスインスタンスの挙動全体が完全にそのコンポーネントにカプセル化される。他の場合では（上部の楕円)、コンポーネントは、グリッドサービス挙動の実装を完全なものとするために、他のサーバーサイドのプロセスと協調するだろう。この際、コンポーネントとサーバーサイドプロセスとの協調は、おそらくアダプタレイヤを介する。図 3 下部に示したもう 1 つのシナリオでは、ネットワークメッセージのデマーシャル、デコーディングを含むグリッドサービスインスタンスの挙動全体を 1 つのプロセスにカプセル化する。このアプローチには効率上の優位点は存在するかもしれないが、グリッドサービス実装間の機能再利用の機会をほとんど提供しない。

コンテナ実装は、単純な引数デマーシャリング以上の機能を提供するかもしれない。例えば、生存期間管理機能、認可や認証の自動的サポート、リクエスト記録、生存期間管理機能のインターセプト、生存期間が過ぎるか明確な破棄リクエストを受け取った場合のサービスインスタンス破棄などである。従って、我々は異なるグリッドサービス実装でこれらの共通挙動を再実装する必要性を回避する。

4 グリッドサービス

この文章の目的は、**グリッドサービス**を定義するインターフェイスと挙動を明確に述べることである。簡潔に言えば、**グリッドサービス**とは、インターフェイス定義と挙動に関連した規約に従う、WSDL で定義されたサービスである。Web サービスはグリッドサービスではないが、すべての**グリッドサービス**は Web サービスである。以下の節では、次の事柄を詳しく説明していく。

- 我々がグリッドサービス仕様で用いる WSDL 規約を導入する。これらの規約は WSDL 1.2 に取り込まれた [WSDL 1.2 DRAFT]。
- **サービスデータ**を定義する。サービスデータとは、サービスインスタンスのメタデータと状態データを表現するためとそれらに対する問い合わせを行うための、標準的な手段を提供する。
- **グリッドサービス**の一連の核となるプロパティを紹介する。以下はその一部である。
    - **グリッドサービス記述**と**グリッドサービスインスタンス**を、拡張や使用を構造化する原理として定義する
    - OGSI が時間をモデル化する方法を定義する。
    - グリッドサービスハンドルとグリッドサービス参照を定義する。これらは、グリッドサービスインスタンスを参照するために用いられる。
    - 操作に関する故障情報を通知する共通のアプローチを定義する。このアプローチは、共通の解釈をサポートするために、WSDL の故障メッセージに対して基本となる XML スキーマ定義と関連した意味を定義する。このアプローチは、WSDL の故障メッセージモデルを変更しないで、単に故障メッセージの基本となる形式を定義する。
    - グリッドサービスインスタンスのライフサイクルを定義する。

5 WSDL 拡張と規約

OGSI は、Web サービスに基づている。特に、OGSI は WSDL をグリッドサービスの公開インターフェースを記述する機構として使用する。しかしながら、WSDL1.1 は 2 つの重要な点において不完全である。1 つ目は、インターフェース(portType)の拡張が不可能なことであり、2 つ目は、ポートタイプについて追加の情報を記述不可能なこと(オープンコンテントの欠如)である。これらの問題は W3C Web Services Description Working Group で取り組まれてきた[WSDL 1.2 DRAFT]。しかし、WSDL1.2 の策定は「進行中」であるため、OGSI は WSDL1.2 全体を直接取り込むことはできない。

そうではなく、OGSI は、wsdl:portType 要素とは別に WSDL1.1 に対しての拡張を定義する。同拡張は、WSDL1.1 に対して必要とされる最小限の拡張を提供する。これらの WSDL1.1 に対する拡張は、W3C Web Services Description Working Group によって合意された機能と等価である。WSDL 1.2 [WSDL 1.2] が W3C 勧告として公開された後に、Global Grid Forum は必ず WSDL 1.2 を用いた OGSI の次バージョンを定義し、OGSI v1.0 で定義する拡張から WSDL 1.2 への変換を定義する。

我々は、WSDL1.1 に対する変更を明確にするために、接頭辞が gwsdl である、分離された一時的な名前空間を定義する。特に、gwsdl はオープンコンテントモデルと portType 拡張をサポートするために、[WSDL 1.2 DRAFT] で提案された方法によって wsdl:portType 要素に以下の新たな構文を追加する。

以下が gwsdl:portType の XSD 定義である。

```
…
targetNamespace=
   http://www.gridforum.org/namespaces/2003/03/gridWSDLExtensions
xmlns:gwsdl=
   http://www.gridforum.org/namespaces/2003/03/gridWSDLExtensions
…

<element name="portType" type="gwsdl:portTypeType"/>
<complexType name="portTypeType">
   <complexContent>
      <extension base="wsdl:portTypeType">
         <sequence>
            <any namespace="##other"
                 minOccurs="0" maxOccurs="unbounded"/>
         </sequence>
         <attribute name=”extends” use=”optional”>
            <simpleType>
               <list itemType=”QName”/>
            </simpleType>
         </attribute>
         <anyAttribute namespace="##other"/>
      </extension>
   </complexContent>
</complexType>
```

gwsdl:portType 要素の **extends** 属性は QName のリストであり、それぞれの QName は wsdl:portType か gwsdl:portType のどちらかを参照**しなければならない(MUST)**。

portType 拡張について考慮すると、operation 要素の名前空間に関して明確にするべき点がある。WSDL 1.1 は operation 要素の名前のオーバーロードを許しているが（WSDL 1.1 portType は、同名の複数の operation 子要素を持つことが可能)、W3C Web Services Description Working Group はこれを制限することを決定した。以下は[WSDL 1.2 DRAFT]からの抜粋である。

> ポートタイプ操作コンポーネントは、あるポートタイプがサポートする操作を説明するものである。操作とは、メッセージ参照の集合である。メッセージ参照はこの操作が受理するメッセージ（すなわち入力メッセージ)、またはこの操作が送信するメッセージ（すなわち出力や故障メッセージ）を参照する。
>
> ポートタイプ操作コンポーネントは、{name}と{target namespace}プロパティ両方を持つにもかかわらず、ポートタイプコンポーネントにローカルなものであり、Qname で参照することはできない。

ポートタイプ操作コンポーネントのプロパティは以下の通りである。

- {name} [XML Namespaces] で定義される NCName。
- {target namespace} [XML Namespaces]で定義される名前空間名。
- {variety} Input-Only、Output-Only、Input-Output、Output-Input のどれか 1 つ。
- {message} メッセージ参照コンポーネントの集合。

あるポートタイプの {operatoins} プロパティ内のポートタイプ操作コンポーネントそれぞれについて、{name}と{target namespace}プロパティの組み合わせはユニークでなければならない。

1 つ以上のポートタイプを拡張するポートタイプのために、2 つ以上のポートタイプ操作コンポーネントが{name}と{target}プロパティについて同じ値を持つ場合は、これらのポートタイプ操作コンポーネントのコンポーネントモデルは等価で**なければならない(MUST)**(2.12 コンポーネントの等価性を参照)。もし、ポートタイプ操作コンポーネントが等価なとき、それらは単一のコンポーネントにまとめられるとみなされる。もし、2 つのポートタイプ操作コンポーネントが{name}と{target namespace}プロパティについて等しい値を持つが、それらのコンポーネントが等価でない場合は、エラーである。

注意:

上記規則のために、{target namespace}プロパティについて同じ値を持つ 2 つのポートタイプが、{name}プロパティが同じ値を持つ 1 つ以上の操作を持つ場合を考える。この時、それらの操作が同じ操作でない限り、その 2 つのポートタイプが共に 1 つの派生ポートタイプの派生チェーンの一部とはなりえない。それゆえ、1 つの名前空間の操作名をユニークにすることを必要な時に保証することは、そのような派生がエラーを起こさないことになり、良い習慣とみなされる。

以下は、コンポーネントの等価性についての引用である。

同じ型の 2 つのコンポーネントについて、1 つ目のコンポーネントのプロパティ値が 2 つ目のコンポーネントのプロパティ値と同じならば、それらは等価であるとみなされる。

トップレベルコンポーネント（メッセージ、ポートタイプ、バインディング、サービス）に関しては、これは、名前に関する制約を与えられた、名前を基にした等価性に事実上変換される。すなわち、同じ型の 2 つのトップレベルコンポーネントを与えられて、{name}プロパティと{target namespace}プロパティ共に同じ値を持つならば、それら 2 つのコンポーネントは実際に同じコンポーネントである。

gwsdl portType における、操作の名前付けと拡張は、上記 WSDL 1.2 DRAFT によって定義される方法と同じく解釈されるべきである。

6 サービスデータ

OGSI で導入された**状態を有する** Web サービスを実現するアプローチにおいて、クエリ、更新、変更通知のためにサービスインスタンスの状態をサービスの要求者へ示す、共通の機構が必要であることがわかった。この考えは、グリッドアプリケーション以外のコンテキストにおける Web サービスにも適用可能なため、我々は "serviceData" と呼ばれる Web サービスの状態を表すデータを外部に公開する共通のアプローチを提案する。我々はこの考えをより広範囲な Web サービスコミュニティに紹介していく。

状態を有する Web サービス（すなわち、**グリッドサービス**）のインターフェイスを完全に説明するために、その状態の外部から観察可能な要素について説明する必要がある。ここで、外部から観察可能とは、サービスインスタンスの状態が宣言されたサービスインターフェイスを用いてクライアントに公開されていることを意味する（クライアントは、サービスインスタンス自身の実装の外部に存在するものを意味する）。サービスデータをサービスの外部インターフェイスの一部として宣言する必要性は、オブジェクト指向インターフェイス定義言語（IDL）において、属性をオブジェクト指向インターフェイスの一部として宣言するという考えとほぼ同じものある。サービスデータは読み込み、更新、登録要求に対して公開されうる。

WSDL は portType に対する操作とメッセージを定義するため、宣言されたサービス状態はサービスインターフェイスの一部として定義されたサービス操作を通してのみ、外部アクセス可能で**なければならない (MUST)**。serviceData に固有な操作をそれぞれの serviceData 要素に対して定義する必要性を避けるために、グリッドサービス portType (§9) は、serviceData 要素を名前で操作するための基本操作を提供する。

一例を挙げる。インターフェイス foo は操作 op1、op2、op3 を定義する。また foo インターフェイスは外部アクセス可能なデータとして de1、de2、de3 を持つ。ここで、WSDL を用いて foo と foo に対する操作を表現する。さらに、OGSI serviceData 構文は WSDL を拡張し、設計者が foo の状態 de1、de2、de3 の一部のアクセス制限を宣言することによって、foo へのインターフェイスを定義可能にする。この宣言は、foo インターフェイスを実装する、状態を有するサービスインスタンスのサービスデータに対する操作の実行を容易にする。

単純に言うと、serviceData 宣言はサービスインターフェイスによって外部アクセス可能な状態を表現するために用いられる機構である。serviceData 要素は、本仕様で定義されるようなサービスインターフェイスを通してアクセス可能である。サービスインスタンスのプライベートな内部状態はサービスインターフェイスに含まれず、ゆえに serviceData 宣言で表現されることはない。

6.1 動機と JavaBean のプロパティとの比較

OGSI 仕様は、Web サービスの状態にアクセスする柔軟でプロパティスタイルな方法を提供するために、serviceData という概念を導入する。serviceData という概念は、Java™, Smalltalk、C++のようなオブジェクト指向プログラミング言語における public なインスタンス変数やフィールドといったものと同様なものである。

serviceData は JavaBean™のプロパティに似たものである。JavaBean モデルでは、プロパティにアクセスするメソッドのシグネチャ（getXXX/setXXX）に関する規約と、プロ

パティを文書化するヘルパークラス（BeanInfo）についての規約を定義している。OGSI モデルは、同様の結果を得るために、serviceData 要素と XML スキーマタイプを用いる。

OGSI 仕様では、serviceData 要素に対して getXXX、setXXX といった WSDL 操作を必要としないと選択した。ただし、サービス実装者はそのような型安全な get/set 操作を自身で定義**可能である（MAY）**。OGSI では、そのような操作ではなく、クエリ(get）、更新(set)、serviceData 要素の変更通知登録のための拡張可能な操作を定義する。これらの操作によって、serviceData 要素にサービスインスタンスとの相対的な名前でアクセスすることを可能にする単純な構文が提供される。この名前によるアクセスは、JavaBean や Enterprise JavaBean プログラマにとって馴染み深い、getXXX、setXXX による方法とほぼ等価な機能を提供する。しかし、これらの OGSI で定義される操作は、他のサービスインターフェイスによって、より高機能なクエリ、更新、登録（例えば、1 つのサービスインスタンスに属する複数の serviceData 要素にまたがるクエリなど）をサポートするために拡張**可能である（MAY）**。.

GridService portType 定義における serviceDataName 要素は、JavaBeans における BeanInfo クラスに対応する。しかし、OGSI は serviceData についての情報を提供するために、BeanInfo モデルにおけるシリアライズ可能な実装クラスを用いるのではなく、XML（WSDL）文書を選択した。

6.2 serviceData による portType の拡張

serviceData は、serviceData という名前の新しい portType 子要素を定義する。この portType は関連付けられた serviceData 要素、すなわち SDE を定義するために用いられる。これらの serviceData 要素定義は、serviceData 宣言、または SDD と呼ばれる。（“static” とマークされた）serviceData 要素の初期値は、portType の staticSerivceDataValue 要素で指定**可能である（MAY）**。portType 中で静的に宣言されたものか、Web サービスインスタンスとしての生存期間中に代入されたものであるかに関わらず、serviceData 要素の値は、serviceData 要素値、または SDE 値と呼ばれる。

以下で、gwsdl:portType という記述に注意せよ。これにより、他の名前空間の要素が使用可能になる。

```
   <gwsdl:portType name="NCName"> *
      <wsdl:documentation … /> ?
      <wsdl:operation name="NCName"> … </wsdl:operation> ?
…
      <sd:serviceData name="NCName" … /> *
      <sd:staticServiceDataValues>?
         <some element>*
      </sd:staticServiceDataValues>
…
   </gwsdl:portType>
```

例えば、以下の portType 宣言は 2 つの serviceData 要素を宣言しており、それらの修飾名は”tns:sd1” と “tns:sd2” である。この portType を実装するサービスインスタンスは、その状態の一部としてこれらの serviceData 要素を**必ず含まなければならない（MUST）**。

```
<wsdl:definitions xmlns:tns=″xxx″ targetNamespace=″xxx″>
   <gwsdl:portType name="exampleSDUse"> *
      <wsdl:operation name=…> … </wsdl:operation>
…
      <sd:serviceData name="sd1" type=″xsd:String″
                      mutability=″static″/>
      <sd:serviceData name="sd2" type=″tns:SomeComplexType″/>
…
      <sd:staticServiceDataValues>
         <tns:sdl>initValue</tns:sd1>
      </sd:staticServiceDataValues>
   </gwsdl:portType>
…
</wsdl:definitions>
```

6.2.1 serviceData 宣言の構造

sd:serviceData 宣言は、sd:serviceDataValue 要素と sd:staticServiceDataValue 要素中に出現しうる XML 要素を定義することであるため、sd:serviceData の XML スキーマ定義は xsd:element のスキーマ定義をモデルとしている。sd:serviceData 要素は、xsd:element の場合と同じ意味で 5 つの属性（name、type、minOccurs、maxOccurs、nillable）を用いる。その他の xsd:element の属性は、sd:serviceData 要素では使用しない。sd:serviceData 要素はまた、mutability と modifiable の 2 つの属性を持つことが可能である。

sd:serviceData の XML スキーマ定義は以下の通りである。

```
…
   targetNamespace =
     ″http://www.gridforum.org/namespaces/2003/serviceData″
   xmlns:sd =
     ″http://www.gridforum.org/namespaces/2003/03/serviceData″
…

<attributeGroup name="occurs">
   <attribute name="minOccurs"
              type="nonNegativeInteger"
              use="optional"
              default="1"/>
   <attribute name="maxOccurs">
      <simpleType>
         <union memberTypes="nonNegativeInteger">
            <simpleType>
               <restriction base="NMTOKEN">
                  <enumeration value="unbounded"/>
               </restriction>
            </simpleType>
         </union>
      </simpleType>
   </attribute>
```

```
</attributeGroup>

<complexType name="ServiceDataType">
   <sequence>
      <any namespace="##any" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="name" type="NCName"/>
   <attribute name="type" type="QName"/>
   <attribute name="nillable"
              type="boolean"
              use="optional"
              default="false"/>
   <attributeGroup ref="sd:occurs"/>
   <attribute name="mutability" use="optional" default="extendable">
      <simpleType>
         <restriction base="string">
            <enumeration value="static"/>
            <enumeration value="constant"/>
            <enumeration value="extendable"/>
            <enumeration value="mutable"/>
         </restriction>
      </simpleType>
   </attribute>
   <attribute name="modifiable" type="boolean" default="false"/>
   <anyAttribute namespace="##other" processContents="lax"/>
</complexType>

<element name="serviceData" type="sd:ServiceDataType"/>

<xsd:complexType name=”ServiceDataValuesType”>
   <xsd:sequence>
      <xsd:any namespace="##any" minOccurs="0"
                                 maxOccurs="unbounded" />
   </xsd:sequence>
</xsd:complexType>

<element name=”serviceDataValues”
         type=”sd:ServiceDataValuesType”/>

<element name=”staticServiceDataValues”
         type=”sd:ServiceDataValuesType”/>
```

serviceData の属性は以下の通りである。

- **maxOccurs = (nonNegativeInteger | unbounded) : default to 1**
  - **この値は、サービスインスタンスの serviceDataValues 中または portType staticSerivceDataValues 中の、serviceData 要素値の最大出現回数を表す。**
- **minOccurs = nonNegativeInteger : default to 1**
  - **この値は、サービスインスタンスの serviceDataValues 中または portType staticSerivceDataValues 中の、serviceData 要素値の最小出現回数を表す。**

    - 値が 0 ならば、この serviceData 要素は任意である。
- name = NCName と{ターゲット名前空間}
    - serviceData 要素の名前は、wsdl:definitions 要素のターゲット名前空間中のすべての sd:serviceData 宣言と xsd:element 宣言において、ユニークでなければならない (MUST)。
    - serviceData 要素の名前と wsdl:definitions 要素の targetNamespace 属性が示すターゲット名前空間の組み合わせは、QName を構成し、この serviceData 要素へのユニークな参照を可能にする。
- nillable = boolean : default to false
    - serviceData 要素が値として nil を持ちうるかどうか (value="true"である xsi:nil 属性を持ちうるかどうか) を示す。
        - 例えば、以下のような serviceData 宣言ならば、

```
<serviceDataElement name="foo" type="xsd:string" nillable=true" />
```

        - 以下のように SDE 値を持ちうる。

```
<foo xsi:nil="true"/>
```

- type = QName
    - serviceData 要素値の XML スキーマタイプを定義する。
- modifiable = "boolean" : default to false
    - 真ならば、要求者が setServiceData 操作によって serviceData 値を直接更新することが許可されている (§9.2.2)。ただし、minOccurs と maxOccurs によるカージナリティや以下の mutability の制限には従わなければならない。偽ならば、serviceData 要素は要求者からは "read only" となる。しかし、その値は他のサービスインターフェイスの操作の結果として変化するかもしれない。
- mutability = "static" | "constant" | "extendable" | "mutable" : default to extendable
    - serviceData 要素の値が変化しうるかどうか、また変化するとしたらどのように変化するのかを示す (§6.2.3 を参照)。
- {スキーマの名前空間に属さない任意の属性}
    - serviceData 宣言の属性に関するオープンコンテントを記述可能である。
- コンテント
    - これはオープンコンテント要素モデルである。すなわち、(serviceData 以外の)任意の他の名前空間に属する要素は serviceData 要素の子要素として出現可能であることを意味する。

6.2.2 serviceData の使用: GridService portType による例

serviceData をどのように使用できるかを示すために、第 9 節で説明される GridService portType による例を示す。GridService portType に対して宣言された (非正規な) serviceData 要素は以下の通りである。

```
<wsdl:definitions …

<gwsdl:portType name="GridService" …>
   <wsdl:operation name=…> … </wsdl:operation>
   …
```

```
   <sd:serviceData name="interface" type="xsd:QName"
        minOccurs="1" maxOccurs="unbounded"
        mutability="constant"/>
   <sd:serviceData name="serviceDataName" type="xsd:QName"
        minOccurs="0" maxOccurs="unbounded"
        mutability="mutable" nillable="false"/>
   <sd:serviceData name="factoryHandle"
        type="ogsi:HandleType"
        minOccurs="1" maxOccurs="1"
        mutability="constant" nillable="true"/>
   <sd:serviceData name="gridServiceHandle"
        type="ogsi:HandleType"
        minOccurs="0" maxOccurs="unbounded"
        mutability="extendable"/>
   <sd:serviceData name="gridServiceReference"
        type="ogsi:ReferenceType"
        minOccurs="0" maxOccurs="unbounded"
        mutability="mutable"/>
   <sd:serviceData name="findServiceDataExtensibility"
        type="ogsi:OperationExtensibilityType"
        minOccurs="1" maxOccurs="unbounded"
        mutability="static"/>
   <sd:serviceData name="terminationTime" type="ogsi:terminationTime"
        minOccurs="1" maxOccurs="1"
        mutability="mutable"/>
   <sd:serviceData name="setServiceDataExtensibility"
        type="ogsi:OperationExtensibilityType"
        minOccurs="2" maxOccurs="unbounded"
        mutability="static" />
…
```

個々の serviceData 要素の正規の記述は、第 9.1 節で挙げる。

以下は、グリッドサービスインスタンスに対する serviceData 要素値の例である。

```
…
   xmlns:crm="http://gridforum.org/namespaces/2002/11/crm"
   xmlns:tns="http://example.com/exampleNS"
   xnlns="http://example.com/exampleNS">

<sd:serviceDataValues>
   <ogsi:interface>crm:GenericOSPT</ogsi:interface>
   <ogsi:interface>ogsi:GridService</ogsi:interface>

   <ogsi:serviceDataName>ogsi:interface
   </ogsi:serviceDataName>
   <ogsi:serviceDataName>ogsi:serviceDataName
   </ogsi:serviceDataName>
   <ogsi:serviceDataName>ogsi:factoryHandle
   </ogsi:serviceDataName>
```

```
   <ogsi:serviceDataName>ogsi:gridServiceHandle
   </ogsi:serviceDataName>
   <ogsi:serviceDataName>ogsi:gridServiceReference
   </ogsi:serviceDataName>
   <ogsi:serviceDataName>ogsi:findServiceDataExtensibility
   </ogsi:serviceDataName>
   <ogsi:serviceDataName>ogsi:terminationTime
   </ogsi:serviceDataName>
   <ogsi:serviceDataName>ogsi:setServiceDataExtensibility
   </ogsi:serviceDataName>

   <ogsi:factoryHandle>someURI</ogsi:factoryHandle>

   <ogsi:gridServiceHandle>someURI</ogsi:gridServiceHandle>
   <ogsi:gridServiceHandle>someOtherURI</ogsi:gridServiceHandle>

   <ogsi:gridServiceReference>…</ogsi:gridServiceReference>
   <ogsi:gridServiceReference>…</ogsi:gridServiceReference>

   <ogsi:findServiceDataExtensibility
         inputElement="ogsi:queryByServiceDataNames" />

   <ogsi:terminationTime after="2002-11-01T11:22:33"
                         before="2002-12-09T11:22:33"/>

   <ogsi:setServiceDataExtensibility
         inputElement="ogsi:setByServiceDataNames" />
   <ogsi:setServiceDataExtensibility
         inputElement="ogsi:deleteByServiceDataNames" />

</sd:serviceDataValues>
```

6.2.3 Mutability

serviceData 要素宣言の mutability 属性は、serviceData 要素の値が生存期間中にどのように変化するかを示す。

- mutability="static": SDE 値が WSDL 宣言（staticServiceDataValues）で代入され、portType のすべてのインスタンスにおいてその値を保持**しなければいけない (MUST)** ことを示す。"static" な SDE とは、プログラミング言語におけるクラスメンバー変数と似たものである。
- mutability="constant": SDE 値がグリッドサービスインスタンスの生成時に代入され、そのインスタンスの生存期間中変化**してはいけない (MUST NOT)** ことを示す。
- mutability="extendable": 一度ある要素が SDE 値の一部となったならば、そのグリッドサービスインスタンスの生存期間中はその SDE 値の一部として存在**しなければならない (MUST)** ことを示す。SDE 値に新たな要素を追加することは可能**だが (MAY)**、以後削除は**不可能である (MUST NOT)**。

- mutability="mutable": 任意の時点で、SDE 値中の任意の要素が削除**可能 (MAY)**、かつ他の要素を SDE 値に追加**可能 (MAY)** であることを示す。

注意: 以上の機能は XML スキーマ要素定義における "fixed" と "default" 属性とは異なる。XML スキーマの "fixed" 属性は "static" な値を示すのに使えるかもしれないが、"extendable" な場合と "mutable" な場合は、mutability 属性を用いなければならないだろう。mutability="constant" な場合は、値が代入されたあと変化しないプロパティを記述するのに用いられるだろう。しかし、その値はサービス記述で代入されるのではなく、実行時に初期化されなければならない。

6.3 serviceDataValues

個々のサービスインスタンスは serviceData 要素の集まりと関連づけられる。その serivceData 要素は、サービスインターフェイスを構成する様々な portType 内で定義される要素や、(潜在的には) 実行時に追加される補足的な要素 (§6.5 を参照) である。我々は、サービスインスタンスと関連付けられた serviceData 要素の集合を "serviceData セット" と呼ぶ。serviceData セットはまた、portType インターフェイス階層で宣言されたすべての serviceData 要素から統合される serviceData 要素の集合を参照してもよい (§6.4 を参照)。

個々のサービスインスタンスは、serviceData 要素値を含む serviceDataValues がルート要素である「論理的な」XML 文書を持た**なければならない (MUST)**。我々は、serviceDataValues 要素の例を既に示した。サービスの実装は、どのように SDE 値を保持するかについては自由に選択可能である。例えば、SDE 値を XML としてではなく、XML に変換されるインスタンス変数としてや、または他の何らかの符号化方式によって保持してもよい。

通常、serviceData 要素を扱う様々な操作と関連付けられた wsdl:binding は、サービス要求者とサービス提供者の間におけるデータの符号化方式を示す。例えば、ある binding は、serviceData 要素の値がシリアライズされた Java オブジェクトとして符号化されていると示すかもしれない。

6.3.1 portType の SDE 初期値の定義

portType は、staticServiceDataValues 要素を用いて、任意の mutability が "static" である serviceData 要素の初期値を serviceData セット内で宣言**可能である (MAY)**。これは、serviceData 要素がその portType 内で宣言されているか、その portType が拡張した portType 内で宣言されているかには関係ない。さらに、初期値の個数がその要素の maxOccurs 属性による制限を越えない限り、初期値はインターフェイス階層内の複数の portType で宣言されていても**構わない (MAY)**。この場合、この serviceData 要素の初期値はすべての初期値の集まりとなる。

例えば、以下の宣言は正しいものであり、tns:otherSD の二つの初期値、"initial value 1" と "initial value 2" を宣言する。

```
<wsdl:definitions xmlns:tns="xxx" targetNamespace="xxx">
   <gwsdl:portType name="otherPT">
      <wsdl:operation name=…> … </wsdl:operation>
…
      <sd:serviceData name="otherSD" type="xsd:String"
```

```
                    mutability="static" maxOccurs="unbounded"/>

      <sd:staticServiceDataValues>
         <tns:otherSD>initial value 1</tns:otherSD>
      </sd:staticServiceDataValues>
…
   </gwsdl:portType>

   <gwsdl:portType name="exampleSDUse" extends="tns:otherPT">
      <wsdl:operation name=…> … </wsdl:operation>
…
      <sd:serviceData name="sd1" type="xsd:String"
                    mutability="static" />
      <sd:serviceData name="sd2" type="tns:SomeComplexType"/>

      <sd:staticServiceDataValues>
         <tns:sd1>an initial value</tns:sd1>
         <tns:otherSD>initial value 2</tns:otherSD>
      </sd:staticServiceDataValues>
…
   </gwsdl:portType>
…
</wsdl:definitions>
```

mutability が "static" ではない serviceData 要素の初期値を宣言**してはいけない (MUST NOT)**。

6.4 portType インターフェイス階層における SDE の統合

WSDL 1.2 は、複数 portType 拡張という概念を導入し、我々はその構文を gwsdl 名前空間を用いてモデル化した。portType は 0 個以上の他の portType を拡張可能である。ある wsdl:service と、その WSDL シンタックスでモデル化されたサービスによってサポートされる portType の間には直接的な関係はない。むしろ、そのサービスによって実装される portType の集合は、その service 要素の port 子要素と、その port 要素から参照される binding 要素から導き出されるものである。この portType の集合とそれらが拡張するすべての portType が、サービスの完全なインターフェイスを定義する。

あるサービスインスタンスによって実装される完全なインターフェイスを構成する portType について考える。このサービスのインターフェイスによって定義される serviceData セットは、この portType それぞれで宣言される serviceData 要素の和集合をとったものである。serviceData 要素は QName でユニークに識別されるため、和集合をとることは serviceData 要素が集合中に一度のみ出現可能であることを意味する。例えば、"pt1" という名前の portType と "pt2" という名前の portType が共に "tns:sd1" という名前の serviceData を宣言し、"pt3" という名前の portType が "pt1" と "pt2" 両方を拡張しているとする。この場合、"pt3" portType は"tns:sd1" serviceData 要素を 1 つだけ持つ (2 つではなく)。

以下の例について考える。

```
<gwsdl:portType name="pt1">
```

```
   <sd:serviceData name="sd1" … />
</gwsdl:portType>

<gwsdl:portType name="pt2" extends="pt1">
   <sd:serviceData name="sd2" … />
</gwsdl:portType>

<gwsdl:portType name="pt3" extends="pt1">
   <sd:serviceData name="sd3" … />
</gwsdl:portType>

<gwsdl:portType name="pt4" extends="pt2 pt3">
   <sd:serviceData name="sd4" … />
</gwsdl:portType>
```

上記 4 つの portType で定義される serivceData セットは以下のようになる。

サービスが実装する portType	その servcieData セット
Pt1	sd1
Pt2	sd1, sd2
Pt3	sd1, sd3
Pt4	sd1, sd2, sd3, sd4

6.4.1 portType インターフェイス階層における静的な SDE の初期値

静的な SDE の初期値は、portType インターフェイス階層に従って統合可能である。しかし、カージナリティに関する要求（minOccurs と maxOccurs）は満たされ**なければならない (MUST)**。例えば、以下では、pt1 を実装するサービスインスタンスは sd1 という名前の SDE について、値 <sd1>1</sd1>を持つ。

```
<gwsdl:portType name="pt1">
   <sd:serviceData name="sd1" minOccurs="1" maxOccurs="1"
                          mutability="static"/>
   <sd:staticServiceDataValues>
      <sd1>1</sd1>
   </sd:staticServiceDataValues>

</gwsdl:portType>
```

しかし、以下では、pt2 を実装するサービスインスタンスは sd1 という名前の SDE から値 <sd1>1</sd1> を継承し、かつ sd2 という名前の SDE の値として <sd2>2</sd2> を持つ。

```
<gwsdl:portType name="pt2" extends="pt1">
   <sd:serviceData name="sd2" minOccurs="1" maxOccurs="1"
                          mutability="static"/>
   <sd:staticServiceDataValues>
      <sd2>2</sd2>
```

```
   </sd:staticServiceDataValues>
</gwsdl:portType>
```

pt3 を実装するサービスインスタンスは、sd3 という名前の SDE について <sd3>3a</sd3> と <sd3>3b</sd3> の 2 つの値を持つ。もちろん、このサービスインスタンスは sd1 という名前の SDE の値を継承する。

```
<gwsdl:portType name="pt3" extends="pt1">
   <sd:serviceData name="sd3" minOccurs="1" maxOccurs="unbounded"
                           mutability="static"/>
   <sd:staticServiceDataValues>
      <sd3>3a</sd3>
      <sd3>3b</sd3>
   </sd:staticServiceDataValues>
</gwsdl:portType>
```

pt4 を実装するサービスインスタンスは、pt1 で定義される sd1 の値を継承するが、staticSeviceDataValues 要素が無いことは、sd4 については初期値がないことを意味する（たいていの場合は、pt4 を拡張する portType においてなんらかの初期値が定義されるものだが)。

```
<gwsdl:portType name="pt4" extends="pt1">
   <sd:serviceData name="sd4" minOccurs="0" maxOccurs="unbounded"
                           mutability="static"/>
</gwsdl:portType>
```

pt5 を実装するサービスインスタンスは作成不可能である。sd5 の初期値がなく、minOccurs 値が 0 より大きいため、インスタンスが作成される時にエラーが生成される。この種の portType は、「抽象」portType を宣言したいという設計者の意図がある場合に見受けられる。その抽象 portType を拡張する portType が、minOccurs 属性が 0 より大きな値である、具体的な SDE を定義する。

```
<gwsdl:portType name="pt5" extends="pt1">
   <sd:serviceData name="sd5" minOccurs="1" maxOccurs="unbounded"
                           mutability="static"/>
</gwsdl:portType>
```

pt6 を実装するサービスインスタンス は作成不可能である。この portType は sd1 という名前の SDE に対してさらに値を宣言しているが、sd1 SDE の maxOccurs 値を超えるため、インスタンスが作成される時にエラーが生成される。

```
<gwsdl:portType name="pt6" extends="pt1">
   <sd:staticServiceDataValues>
      <sd1>6</sd1>
   </sd:staticServiceDataValues>
</gwsdl:portType>
```

pt7 を実装するサービスインスタンスが持つ serviceData 要素値の集合は興味深い。その集合は、まず、sd1 SDE について単一の値 <sd1>1</sd1> を持つ。pt2 と pt3 を通して pt1 を継承するにもかかわらず、sd1 の初期値は 1 つだけである。<sd2>2</sd2> が、pt2 から継承された、sd2 SDE の唯一の値である。pt3 SDE は、<sd3>3a</sd3>、<sd3>3b</sd3> 、<sd3>7</sd3>の 3 つの値を持つ。この内、最初の 2 つは pt3 から継承されたものであり、最後の値はローカルに定義されたものである。さらに、sd7 という名前の SDE の値がローカルに定義されている (<sd7>7</sd7>)。

```
<gwsdl:portType name="pt7" extends="pt2 pt3">
   <sd:serviceData name="sd7" minOccurs="1" maxOccurs="1"
                           mutability="static"/>
   <sd:staticServiceDataValues>
      <sd7>7</sd7>
      <sd3>7</sd3>
   </sd:staticServiceDataValues>
</gwsdl:portType>
```

要約すると、静的な SDE の値は portType インターフェイス階層に従って統合される。もし結果として SDE 値の集合が、SDE のカージナリティ（値の個数が minOccurs 値より小さい、または maxOccurs 値より大きい）に違反するならば、Web サービスインスタンスが作成される時にエラーが報告される。

6.5 動的な serviceData 要素

多くの serviceData 要素はサービスのインターフェイス定義においてごく自然に定義されるが、インスタンスに動的に serviceData 要素を追加や除去することが便利な状況がありえる。そのような更新の実現方法は実装依存であり、例えば、サービスインスタンスは新しいサービスデータ要素を追加する操作を実装しても**よい (MAY)**。

GridService portType (§9) は動的な SDE の使用を示す。これは、"serviceDataName" という名前の serviceData 要素を含み、その要素は現在定義されている serviceData 要素を列挙する。このサービスインスタンスのプロパティは、サービスインターフェイスを定義する GWSDL で定義された、サービスデータ要素の上位集合を返しても良い。なぜならば要求者は、この serviceDataSet が変化するならば subscribe 操作を、現在の serivceDataSet 値を知るためには findServiceData (§9.2.1) 操作を、それぞれ使うことが可能だからである。

7 重要なグリッドサービスプロパティ

我々はここで、全てのグリッドサービスに共通の多くのプロパティと概念について議論する。

7.1 サービス記述とサービスインスタンス

OGSI においてグリッドサービスの**記述**と、グリッドサービスの**インスタンス**とを区別する。

- **グリッドサービス記述**は、クライアントがサービスインスタンスと相互作用する方法を記述する。この記述は特定のインスタンスによらない。WSDL ドキュメ

ントにおいてグリッドサービス記述は、関連付けられた portType（serviceData 宣言を含む)、バインディング、メッセージ、型定義と共に、そのインスタンスの派生 portType 内で具体化される（派生 portType とは、そのサービスについて記述する wsdl:service 要素の port 子要素によって参照される portType。ここで、その portType の参照は port 要素が参照するバインディング要素を通して行われる）。

- グリッドサービス記述は任意個数の**グリッドサービスインスタンス**によって同時に使用可能であり。このインスタンスは以下を満たす。
    - サービス記述が相互作用する方法を記述する状態を含む。
    - 1 つ以上のグリッドサービスハンドルを持つ。
    - そのインスタンスへの 1 つ以上のグリッドサービス参照を持つ。

サービス記述は主に 2 つの目的のために使用される。1 つ目は、サービスのインターフェイス記述として、ツールがクライアントインターフェイスプロキシやサーバのスケルトンなどを自動生成するために用いることができる。2 つ目は、発見のために用いることができる。例えば、特定のサービス記述を実装したサービスインスタンスや、特定のサービス記述に従うインスタンスを生成するファクトリを発見することができる。

サービス記述は、インターフェイスシンタックスと(非常に初歩的であり、非正規的な方法で)意味の両方を表現しなければならない。**インターフェイスシンタックス**は WSDL の portType によって記述される。**意味**は portType に割り当てられた名前を通して推論されることがある。例えば、グリッドサービスを定義する際、0 個以上のユニークに命名された portType を定義することを考える。仕様書において簡潔な意味をそれらの名前のおのおのに関連付けることが可能である。さらに、この関連付けはおそらく将来的にはセマンティック Web や他のより正式な記述によってなされる。これらの名前は、適切な名前を持ったサービスインスタンスとファクトリの探索によって、クライアントが望むセマンティクスを持つサービスを発見するのに使用されうる。また、これらの名前の定義に名前空間を用いることは大域的にユニークな名前を保証する手段を提供する。

7.2 OGSI における時間のモデル化

本仕様書全体を通して分散グリッド上の複数の団体にとって意味を持つ時間を表現する必要性が生じる。例えば、消費者に情報の有効期間を伝達するためにその情報に生産者によってタイムスタンプが付加されることがある。また、クライアントはサービスとサービスインスタンスの生存期間について交渉する必要がある。さらに、クライアントにサービスの同時利用と相互作用の管理を可能とさせるため、多数のサービスは時間に関する共通認識を持つ必要があるだろう。

操作が絶対時間をあいまいさなく参照可能にするために、GMT 世界標準時間がグリッドサービスのために仮定される。だが、時間の表現のために GMT 世界標準時間を仮定することは、グリッド上のクライアントとサーバ間におけるいかなるレベルの時刻同期も示さ**ない**。事実、本仕様書において具体的な同期の精度はそれがサービスの質の問題であるため特定も期待もされない。

グリッドサービスホスティング環境とクライアントは、GMT 世界標準時間にそれらの時刻を同期させるために Network Time Protocol (NTP) や同等機能を利用する**べきである**

(SHOULD)。だが、クライアントやサービスはメッセージに含まれる時刻が不十分な同期のために範囲外の値であっても、そのメッセージを受理し適切に動作し**なければならない(MUST)**。ここで、「適切に」ということはその時間値に関連した情報の使用を拒絶することを含むことが**許されている(MAY)** ことを意味する。さらに、自身の時刻の精度や解像度が許す粒度を超えた大域順序や同期を要求するクライアントとサービスは、トランザクションや調整されたグローバルクロックのような、補足的な同期サービスインターフェイスを用いることによって調整し**なければならない (MUST)**。

ある場合においては、零時刻と無限時刻と両方の表現が要求される (例えば§7.3 と §9.2)。零時刻は、過去の時刻によって表現される**べきである (SHOULD)**。だが、無限時刻は時間概念の拡張を要求する。それゆえ、我々は ogsi 名前空間に下記の型を導入する。この方は特別値 "infinity" が適切な場合、xsd:dateTime の代わりに使用**可能である (MAY)**。

```
…  targetNamespace =
     "http://www.gridforum.org/namespaces/2003/03/OGSI"

<simpleType name="ExtendedDateTimeType">
   <union memberTypes="ogsi:InfinityType xsd:dateTime"/>
</simpleType>

<simpleType name="InfinityType">
   <restriction base="string">
      <enumeration value="infinity"/>
   </restriction>
</simpleType>
```

7.3 XML 要素生存期間宣言のプロパティ

serviceData 要素はサービスインスタンスの動的な状態のある瞬間の情報を表現することができるため、serviceData の消費者がそれらの情報の有効な生存期間を知り得ることは重要である。クライアントは時間関係の情報を、自由に無視することもできるが、serviceData 要素とその値の妥当性と可用性について論じるために使用することが**できる(MAY)**。

我々は、XML 要素やその子要素と関連付けられた生存期間を記述する 3 つの XML 属性を定義する。これらの属性は、serviceData 要素を含む拡張性属性を許す任意の XML 要素において使用**可能である (MAY)**。

3 つの生存期間宣言のプロパティは以下の通りである。

- ogsi:goodFrom: 要素の内容がその時点から有効であるとされる時刻を宣言する。典型的には値が作成された時間である。
- ogsi:goodUntil: 要素の内容がその時点まで有効であるとされる時刻を宣言する。このプロパティは、goodFrom 時刻以降で**なければならない(MUST)**。
- ogsi:availableUntil: 要素自身がその時点まで参照可能であると期待される時間を宣言する。ただし要素の内容自体の更新は起こってもよい。この時刻より前には、クライアントはこの要素の更新されたコピーを入手可能である**べきである(SHOULD)**。この時間の後には、クライアントはもはやこの要素のコピーを得

ることは不可能であってよい(MAY)。このプロパティは goodFrom 時間以降でなければならない(MUST)。

ogsi 名前空間におけるこれらの属性の XML 定義は以下の通りである (ExtendedDateTimeType の定義は§7.2 で与える)。

```
targetNamespace = "http://www.gridforum.org/namespaces/2003/03/OGSI"

<xsd:attribute name="goodFrom" type="ogsi:ExtendedDateTimeType"/>
<xsd:attribute name="goodUntil" type="ogsi:ExtendedDateTimeType"/>
<xsd:attribute name="availableUntil" type="ogsi:ExtendedDateTimeType"/>

<xsd:attributeGroup name="LifeTimePropertiesGroup">
   <xsd:attribute ref="ogsi:goodFrom" use="optional"/>
   <xsd:attribute ref="ogsi:goodUntil" use="optional"/>
   <xsd:attribute ref="ogsi:availableUntil" use="optional"/>
</xsd:attributeGroup>
```

我々は以下の serviceData 要素の例を生存期間宣言属性を示すためとさらに定義するために用いる。

```
<wsdl:definitions
   targetNamespace="http://example.com/ns"
   xmlns:n1="http://example.com/ns"
   … >

   <wsdl:types>
      <xsd:schema …
         "targetNamespace=http://example.com/ns"
         …>
         <xsd:complexType name="MyType">
            <xsd:sequence>
               <xsd:element name="e1" type="xsd:string" minOccurs="1"/>
               <xsd:element name="e2" type="xsd:string" minOccurs="1"/>
               <xsd:element name="e3" type="xsd:string" minOccurs="0"/>
            </xsd:sequence>
         <anyAttribute namespace="##any"/>
         </xsd:complexType>
      </xsd:schema>
   </wsdl:types>
   …
   <gwsdl:portType name="MyPortType">
      …
      <sd:serviceData name="mySDE" type= "n1:MyType"
                      minOccurs="1" maxOccurs="1"
                      mutability="mutable"/>
      …
   </gwsdl:portType>
   …
</wsdl:definitions>
```

サービスインスタンスの serviceDataValues の中身は以下の通りである。

```
<sd:serviceDataValues
   <n1:mySDE
          goodFrom="2002-04-27T10:20:00.000-06:00"
          goodUntil="2002-04-27T11:20:00.000-06:00"
          availableUntil="2002-04-28T10:20:00.000-06:00">
      <n1:e1>
         abc
      </n1:e1>
      <n1:e2 ogsi:goodUntil="2002-04-27T10:30:00.000-06:00">
         def
      </n1:e2>
      <n1:e3 ogsi:availableUntil="2002-04-27T20:20:00.000-06:00">
         ghi
      </n1:e3>
   </n1:mySDE>
</sd:serviceDataValues>
```

nl:mySDE 要素の goodFrom と goodUntil 属性はその全ての子要素に当てはまる。これらの属性は、この SDE の消費者に要素の値の期待される生存期間を宣言する。この例では 2002 年 4 月 27 日の午前 10:20（EST）から午前 11:20 である。それゆえ、この SDE を午前 10:20 に受け取る消費者は 1 時間後にはこの SDE の値はもはや有効でないであろうことを知らされる。その結果クライアントは、新しい値を手に入れるためにサービスインスタンスを 11:20 に(SDE の同じ名前 nl:mySDE を与えることで）再度問い合わせるだろう。

availableUntil は、SDE のサービスデータ値を言及するのでは**なく**、むしろこの指定された serviceData 要素自身の可用性を言及する。宣言された availableUntil 時間より前には、クライアントは同じグリッドサービスインスタンスにこの指定された SDE の更新された値を問い合わせ可能である**べきである(SHOULD)**。この例では、クライアントは同じサービスインスタンスに 2002 年 4 月 28 日午前 10:20（EDT）までは、n1:mySD という名前の serviceData 要素を問い合わせ、その値の更新されたコピーを返答として受け取ることが可能であるべきである。だがその時間の後にはクライアントはそのような値が参照可能であることを仮定する**べきでない(SHOULD NOT)**。すなわち、そのような問い合わせの結果はそのようなサービスデータ要素が存在しないことを示す返答となるかもしれない。言い換えれば、SDE の消費者はこの指定された SDE の更新された値を 1 日間取得可能であることを期待できるが、その後サービスインスタンスはもはや n1:mySDE という名の SDE を持っていないかもしれないことを知らされるだろう。この可用性の示唆にもかかわらず、SDE はクライアントに知らされている availableUntil に先立って利用できなくなっても**よい(MAY)**。

SDE 値が生存期間が親要素で宣言されるものとは異なる子要素を含むことが望ましい場合がある。この要求を満たすために、serviceData 要素値に含まれる XML 要素は goodFrom、goodUntil、availableUntil 属性のいかなる組み合わせでも利用**できる(MAY)**。但し、その要素のスキーマがこれらの拡張性属性を許すと仮定する。子要素の

この属性は親要素で指定される初期値をオーバーライドする。子要素におけるこれらの属性の値について親要素において指定される値との制約はない。但し、すべての要素で有効な goodFrom、goodUntil、availableUntil 値の間の順序制約は保たれなければならない。

上述の例において、親要素(n1:mySDE)の生存期間属性はその全ての子要素の初期値となる。例えば、上述の n1:e1 要素はこの初期値を用いる。だが、n1:e2 要素は goodUntil 属性をオーバーライドする。このことは、この値 (“def”) は親要素で宣言されている 1 時間の代わりに 10 分間だけ有効であることが求められるということを明示する。この様な状況は、複雑な要素の一部が他の要素の一部よりも早く変化する場合に生じるだろう。同様に n1:e3 要素は availableUntil をオーバライドする。このことは、10 時間後には n1:mySDE 内にはもはや n1:e3 要素は存在しないかもしれないということを明示する。言い換えれば、10 時間後にはこの serviceData 要素の値を問い合わせた要求者には n1:e3 子要素を含まない n1:mySDE 要素が返されることがあっても**よい (MAY)**。当然この例では n1:MyType スキーマは n1:e3 が任意要素であることを許し、それゆえ n1:MyType から省略されうることを仮定している。

ここで、サービスデータ宣言において、型として使用されることを意図された要素の XML スキーマについて考える。細粒度な生存期間宣言を可能にするために、この XML スキーマは、定義する全ての要素をこれらの生存期間宣言プロパティによる拡張可能とすることが**推奨される(RECOMMENDED)**。

もしある要素でこれらの属性が出現しないならば、goodFrom、goodUntil、availableUntil プロパティは未知である。

7.4 インターフェイス命名と変更管理

分散システムの重要な問題の 1 つは長期に渡ってサービスの更新を可能にすることである。この要求は、クライアントがいつサービスがそのインターフェイスや実装を変更したかを判断できる必要性を伴うことを意味する。ここで、我々はこの問題やそれを解決するための OGSI の機構、要求、推奨について議論する。

7.4.1 変更管理問題

ある特定のグリッドサービスインスタンスのセマンティクスは以下の組み合わせで定義される。

1. **インターフェイス仕様**。構文的には、グリッドサービスのインターフェイスは portType、操作、serviceData 宣言、メッセージ、型を含むサービス記述によって定義される。意味的には、インターフェイスは他の正式なアプローチによっても定義されるだろうが、典型的には本文書のような仕様書で定義される。
2. **インターフェイス実装**。意図される実装セマンティクスはインターフェイス仕様により示されるかもしれないが、結局は実装が全てのグリッドサービスインスタンスのセマンティクスを真に定義する。実装の決定と誤りは、その結果としてサービスインスタンスの挙動が明白でなかったりインターフェイス仕様と反することに帰着するかもしれない。それにもかかわらず、そのサービスインスタンスのクライアントはそのような実装セマンティクスに偶然であろうと故意であろうと依存するかもしれない。

クライアントがグリッドサービスを確実に発見、使用可能であるために、クライアントはこれらの 2 つのサービス定義に適合するかどうか判断可能でなければならない。言い換えれば、グリッドサービス記述はクライアントが要求する portType を含むか、ということである。さらに、実装はクライアントが要求する、例えば重要なバグフィックスを含む特定のパッチレベルなどの、セマンティクスを持っているか、ということでもある。

さらに、グリッドサービス記述は長い時間内に必然的に発展する。グリッドサービス記述が後方互換性を保ちつつ拡張されるなら、拡張前のグリッドサービスの定義を要求するクライアントは新しく拡張された記述をサポートするグリッドサービスインスタンスを使用可能だろう。インテーフェイス定義に対するそのような後方互換な拡張があるだろう。例えば、そのインターフェイスへの新しい操作やサービスデータ記述の追加、または既存の操作への任意の拡張の追加などが考えられるだろう。あるいは、バグを修正するパッチなどの実装変更においても後方互換な拡張があるだろう。例えば、操作を失敗させたエラーを訂正する新しい実装は一般的に後方互換であると見なされるだろう。

だが、クライアントはグリッドサービス記述が後方互換で**ない**方法で変更された場合に検知可能で**なければならない(MUST)**。また、後方互換性の欠如はグリッドサービスインターフェイスや実装に対する互換性の無い変更の結果かもしれない。例えば、ユーザが利用することを覚えた「誤りのある」挙動を「修正する」バグ修正は、後方互換であると見なされないだろう。

この議論は、異なるインターフェイスや実装をサポートするグリッドサービスについてあいまいさなく互換性を宣言可能であることに加えて、グリッドサービスのインターフェイスと実装に簡潔な名前を提供可能であることの必要性を示す。

7.4.2 グリッドサービス記述の命名規約

WSDL では、各 portType はその修飾名によって大域的かつユニークに名前付けされる。portType の修飾名とは、その portType 定義を含む名前空間とその名前空間において局所的にユニークな portType 要素の名前の組み合わせである。OGSI において、変更管理について考慮した結果、我々はグリッドサービス記述の全ての要素は不変で**なければならない(MUST)**という結論に辿り着いた。グリッドサービス portType の QName、操作、メッセージ、serviceData 宣言、基礎となる型定義は唯一つの WSDL/XSD 記述を参照すると仮定されることが**許されている(MAY)**。もし変更が必要とされるならば、新しい portType が新しい QName を伴って定義され**なければならない(MUST)**。すなわち、新しい局所名および、または新規名前空間で定義されなければならない。

開発中は、グリッドサービス記述は portType インターフェイスレベルや実装レベルで頻繁に変更されるかもしれないことに注意せよ。上述した強い不変性要求のため、開発者はリリーススケジュールを慎重に決めるべきである。ひとたびインターフェイスや実装が開発者の完全な制御の外部での使用下に置かれると、新しい portType の導入以外にさらなる変更は許されない。

既存の挙動の変更を除く、グリッドサービスの機能を拡張するインターフェイスの変更は、portType 拡張によってモデル化される**べきである (SHOULD)**。なぜならば、既存のクライアントが修正無しで新しいサービスの使用可能だからである。このようなインターフェイス拡張は完全に新しい portType によってモデル化されることも**可能 (MAY)** だが、このアプローチは推奨されない。

7.5 グリッドサービスインスタンスの命名

各グリッドサービスインスタンスは、大域的にかつ全時間的に 1 つ以上の**グリッドサービスハンドル（GSH）**によって命名される。GSH は URI の形態を採った最小限の名前であり、クライアントにサービスインスタンスへ直接通信可能にする十分な情報は持たない。代わりに、サービスインスタンスと通信を望むクライアントは、GSH を**グリッドサービス参照（GSR）**に解決しなければならない。GSR は、1 つ以上のネットワークプロトコルバインディングを通してサービスインスタンスと通信するためにクライアントが要求するすべての情報を含む。

あらゆる URI と同様に、GSH は**スキーム**とそれに続くスキーム固有のデータを含む文字列からなる。スキームは、GSH を GSR に解決するためにスキーム固有のデータを後に定義される要求の制限の範囲内でどのように解釈するかを示す。クライアントは自身で GSH を解決することも**できる(MAY)**。または、例えば HandleResolver portType(§•を参照)を実装した予め設定済みのサービスインスタンスなどに全ての解決を委託することも**できる(MAY)**。

GSR の形式は、クライアントがグリッドサービスインスタンスと通信するために用いられるバインディング機構に固有である。例えば、RMI/IIOP バインディングを用いるときには GSR は IOR の形式を取るだろう。SOAP バインディングを用いるならば、GSR は適切に注釈付けられた WSDL 文書の形式を取るだろう。

GSH はグリッドサービスインスタンスの生存期間全体において有効である一方、GSR は無効になるかもしれない。その結果、クライアントは GSH を新しい有効な GSR に解決しなければならない。

7.5.1 グリッドサービス参照（GSR）

グリッドサービスインスタンスは、グリッドサービス参照(GSR)を使用することで（潜在的に遠隔の）クライアントアプリケーションからアクセス可能になる。GSR は、典型的には、実行環境に配置された、特定のグリッドサービスインスタンスへのネットワークワイドなポインタであると言える。クライアントアプリケーションは、GSR によって同定される、特定の(潜在的にネットワーク接続された)サービスエンドポイントに存在する、特定のインスタンスに(目的のサービスインスタンスの WSDL portType で定義される操作によって表現される)リクエストを直接送信するために GSR を使用可能である。言い換えれば、GSR は「参照による」グリッドサービスインスタンスを渡すという、プログラム上の概念をサポートする。GSR は、ホスティング環境に存在するグリッドサービスインスタンスへ 1 つ以上の通信プロトコルバインディングを通じてアクセスするために要求される全ての情報を含む。だが、GSR は与えられたクライアントコンテキストやホスティング環境に局所化されるかもしれない。GSR の可搬性のスコープは、それがサポートするバインディング機構によって決定される。GSR の局所化形式（localization format）は、バインディング層でサービス間の参照渡しが検出可能であるような十分なスコープ情報を含む**べきである（SHOULD）**。例えば、局所化されたスコープ外で GSR を使用することが、間違ったサービスインスタンスへの誤ったバインディングとなる**べきではない(SHOULD NOT)**。

グリッドサービス参照のエンコーディングはシステムにおいて様々な形態を採りうる。他の操作メッセージ部と同様に、実際の GSR の符号化された「転送中の」形式は、クラ

イアントによってグリッドサービスインスタンスとの通信に用いられる Web サービスバインディング機構特有のものである。以下でいくつかのバインディングで用いることが**できる (MAY)** GSR の WSDL エンコーディングを定義するが、特定のエンコーディングの使用法はバインディング仕様で定義されるため本仕様の範囲外である。

それでもやはり、この点について更に詳しく述べることは有益である。例えば、RMI/IIOP バインディングが用いられるならば、GSR は CORBA 準拠 IOR として符号化されるだろう。SOAP バインディングを用いるなら、GSR は以下で定義される WSDL エンコーディングの形式をとるだろう。このグリッドサービス参照の「転送中の」形式は 2 つの場合に作成される。1 つ目は、WSDL で定義された操作の応答として参照が返される時に、グリッドサービスのホスティング環境で作成される場合である。2 つ目は、WSDL で定義された操作の入力引数として参照が渡される時に、クライアントアプリケーションもしくはそれが指示する実行環境によって作成される場合である。ここで、WSDL で定義された操作要求メッセージの引数として渡される、このグリッドサービス参照の「転送上の」形式について考える。この形式は、指定されたサービスインスタンスによってサポートされる任意の通信プロトコルを介した、そのサービスインスタンスとの通信に必要なサービスエンドポイントバインディングアドレス情報の全てを含むべきである (SHOUD)。これは、WSDL で定義された操作要求メッセージを運ぶために用いられる Web サービスバインディングプロトコルに依らない。

ある 1 つのグリッドサービスインスタンスを参照する任意個数のグリッドサービス参照がシステム内に存在しても**よい (MAY)**。GSR のライフサイクルは関連付けられたグリッドサービスインスタンスのライフサイクルに依存しなくても**よい (MAY)**。GSR は、関連付けられたグリッドサービスインスタンスが存在し、かつグリッドサービス参照を用いてアクセス可能である時に有効であると言える。無効な GSR はその GSR の使用を試みるクライアントによって検知可能で**なければならなく (MUST)**、GSR スキームによっては他の手段で検知可能であっても**よい (MAY)**。GSR はグリッドサービスインスタンスの生存期間中に無効になることが**許されている (MAY)**。この事態は典型的にはグリッドサービスホスティング環境にもたらされる変更に起因する。これらの変更には、ホスティング環境でサポートされる Web サービスバインディングプロトコルの変更や、グリッドサービスインスタンス自身の破棄を含み**うる (MAY)**。クライアントによる無効なグリッドサービス参照の使用は、そのクライアントによって検知されるか、もしくはそのクライアントへ投げられる例外となる**べきである (SHOULD)**。

グリッドサービス参照が無効であると分かり、かつそれが指定するグリッドサービスインスタンスが存在するときを考える。この場合は、第 7.5.2 **エラー！参照元が見つかりません。** 節で定義されるように、クライアントはそのグリッドサービスインスタンスのグリッドサービスハンドルを用いて新しい GSR を得ることが**できる (MAY)**。簡便性のために、グリッドサービスハンドルは、グリッドサービス参照のバインディングに固有な実装に含まれても**よい (MAY)**。

グリッドサービス参照のバインディング固有の実装は有効期限を含むことが**可能である (MAY)**。有効期限とは、その GSR を持つクライアントに対して、その GSR がその時刻以前は有効である**べきであり (SHOULD)**、かつその時刻より後には有効であっては**いけない (MAY NOT)** ことを宣言するものである。有効期限後、クライアントはその GSR の使用を試みることは**可能 (MAY)** だが、そのグリッドサービスインスタンスの GSH を用いて新しい GSR を取得する**べきである (SHOULD)**。有効期限は一切の保証をしないが、それに

もかかわらず次の点において有用である。すなわち、クライアントが、処理を継続するために（潜在的に長い時間を必要とする）更新が必要な GSR が無効になる時点まで単純に待つのではなく、都合の良い時（おそらく他の操作と同時）にそれを更新することを可能にするからである。

GSR を所有すること自体は、クライアントにそのグリッドサービスの操作を呼び出す権限を与えない。言い換えれば、GSR はケイパビリティではない。グリッドサービスの操作呼び出しの認証は GSR の所有とは直交した問題であり、他で解決されるべきである。

GSR の XML スキーマ定義は以下の通りである。

```
targetNamespace = “http://www.gridforum.org/namespaces/2003/03/OGSI”

<xsd:element name="reference" type="ogsi:ReferenceType"/>

<xsd:complexType name="ReferenceType" abstract="true">
   <xsd:attribute ref="ogsi:goodFrom" use="optional"/>
   <xsd:attribute ref="ogsi:goodUntil" use="optional"/>
</xsd:complexType>
```

§7.3 で既に定義された 2 つの属性は、参照するサービスインスタンスではなく GSR の期待される生存期間を与える。クライアントは与えられた GSR を用いてその goodFrom の前や goodUntil の後にサービスインスタンスを呼び出す**べきではない(SHOULD NOT)**。

7.5.1.1 GSR の WSDL エンコーディング

GSR を符号化する WSDL 文書は、特定のグリッドサービスインスタンスへの到達法を完全に説明する最小の情報を含むことが**推奨される(RECOMMENDED)**。WSDL GSR は wsdl:definitions 子要素を含まなければならない。wsdl:definitions 要素は正確に 1 つの wsdl:service 要素を含まなければ**ならず(MUST)**、さらに他の要素を含むことが**できる(MAY)**。

WSDL GSR の XML 定義は以下の通りである。

```
targetNamespace = http://www.gridforum.org/namespaces/2003/03/OGSI”

<xsd:complexType name="WSDLReferenceType">
   <xsd:complexContent>
      <xsd:extension base="ogsi:ReferenceType">
         <xsd:sequence>
            <xsd:any namespace=”http://schemas.xmlsoap.org/wsdl/”
                 minOccurs="1" maxOccurs="1" processContents="lax"/>
         </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
```

7.5.2 グリッドサービスハンドル（GSH）

グリッドサービスハンドルは以下のプロパティを持つ。

1. GSH は有効な URI [RFC 2396]で**なければならない (MUST)**。
2. GSH は大域的にかつ全時間において同じグリッドサービスインスタンスを参照**しなければならない (MUST)**。GSH は、それらが同時に存在しようとしまいと、複数のグリッドサービスインスタンスを参照**してはならない (MUST NOT)**。 「同じグリッドサービスインスタンス」の定義は§7.5.2.1 を参照せよ。
3. 1 つのグリッドサービスインスタンスは、少なくとも 1 つの GSH を所有し**なければならない (MUST)**。
4. 1 つのグリッドサービスインスタンスは、異なる URI スキームを用いる複数の GSH を所持することが**できる(MAY)**。
5. 1 つのグリッドサービスインスタンスは、同じ URI スキームを用いた複数の GSH をそのスキームが許すならば所持することが**できる(MAY)**。だが、特定の URI スキームの仕様は、このプロパティに対して 1 つのグリッドサービスインスタンスに対してその URI スキームを用いる GSH をただ 1 つのみ許すという制限をかけても**よい (MAY)**。
6. グリッドサービスインスタンスの gridServiceHandle service data 要素値(§9.1)は、そのインスタンスを参照する GSH のみを含ま**なければならない (MUST)**。もしグリッドサービスインスタンスの gridServiceHandle SDE 値に 2 つ以上の GSH が含まれるならば、それらは同じグリッドサービスインスタンスを参照**しなければならない(MUST)**。gridServiceHandle SDE 値はそのグリッドサービスの全ての GSH の部分集合を含むことが**できる(MAY)**。
7. 同じグリッドサービスインスタンスを参照する複数の GSR が存在しても**よい (MAY)**。
8. 同じ GSH の多数の解決の結果は異なる GSR になり**うる (MAY)**。あるリゾルバは異なる時刻において同じ GSH に対して異なる GSR を返すことが**できる(MAY)**。さらに、同じ GSH を解決する異なるクライアントに対して、異なる GSR を返すことが**できる(MAY)**。
9. GSH は、参照するグリッドサービスインスタンスの存在以前、存在中、存在以後で解決不可能であっても**よい (MAY)**。だが、特定の URI スキームの仕様は解決のためのより協力なサービスの質を定義**できる (MAY)**。例えば、特定の URI スキームは、インスタンス生存期間内やインスタンスが終了した後の信頼できる終了宣言までの期間における解決を保証しても**よい (MAY)**。
10. クライアントのサービスインスタンスへの信頼はクライアントが選択する任意の方法で制御**できる(MAY)**。それゆえ、GSH から GSR への解決は信頼できる操作でも信頼できない操作でも**よく(MAY)**、例えばクライアントの設定や解決プロトコルの定義に依存する。本仕様ではクライアントが GSH から GSR への解決を信頼することを許すが要求はしない。

GSH の XML 定義は以下の通りである。

```
targetNamespace = “http://www.gridforum.org/namespaces/2003/03/OGSI”

<xsd:element name="handle" type="ogsi:HandleType"/>
```

```
<xsd:simpleType name="HandleType">
   <xsd:restriction base="xsd:anyURI"/>
</xsd:simpleType>
```

7.5.2.1 サービスインスタンスの同一性

上述した議論で「GSH は大域的ににかつ全時間において**同じグリッドサービスインスタンスを参照しなければならない(MUST)**」と述べた。この要求は、例えばクライアントに同じサービスインスタンスの複数の呼び出しの意味を判断可能にするため、重要である。だが、「同じグリッドサービスインスタンス」という語句の解釈はサービスの記述のセマンティクスに依ることに注意すべきである。サービスインスタンスは、そのサービス記述に関連付けられたセマンティクス、すなわちサービスインスタンスが実装する portType に従う限りは、任意の方法で実装可能である。

例えば、サービス記述に関連付けられたセマンティクスに従う限り、サービスの実装は多様な資源にまたがって分散もしくは複製されても**よい(MAY)**。単一の GSH がこのサービスインスタンスと関連付けられるだろう。ただしその GSH は、異なる資源を参照する異なる GSR へと解決されうる。この解決は、資源可用性や利用率、クライアントの所在地、クライアントの特権などのような因子に基づいて行われる。サービス記述には、そのような複製された実装間で強い状態一貫性を要求するものもあるだろう。例えば、サービス記述のセマンティクスが、サービスインスタンスが特定の一連のメッセージに応じて明確に定義された状態を推移することを要求するかもしれない。それゆえ、状態の一貫性がどのように GSH が GSR に解決されるかに関わらず要求される。だが、他のサービス記述では、分散サービス実装の様々な構成要素間により弱い一貫性を許すよう定義されるかもしれない。

7.5.3 サービスロケータ

サービスロケータは 0 個以上の GSH、0 個以上の GSR、0 個以上のインターフェイス (portType) Qname を含む構造体である。全ての GSH と GSR は同じグリッドサービスインスタンスを参照**しなければならなく(MUST)**、全てのインターフェイスはそのグリッドサービスインスタンスによって実装され**なければならない(MUST)**。ロケータは、本仕様中の GSH と GSR とのどちらかを受理可能な様々な操作によって用いられる。ロケータの XML 定義は以下の通りである。

```
targetNamespace = “http://www.gridforum.org/namespaces/2003/03/OGSI”

<xsd:element name="locator" type="ogsi:LocatorType"/>

<xsd:complexType name="LocatorType">
   <xsd:sequence>
      <xsd:element ref="ogsi:handle"
                   minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="ogsi:reference"
                   minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element name=”interface” type=”QName”
                   minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
```

7.6 グリッドサービスライフサイクル

任意のグリッドサービスインスタンスのライフサイクルは、そのサービスインスタンスの生成と破棄とで境界が定められる。グリッドサービスが生成、破棄される実際の機構は根本的にはホスティング環境のプロパティであり、それ自体は本仕様書では定義されない。だがそれにもかかわらず本仕様では、クライアントがそれらのライフサイクルイベントと共通の方法で相互作用する方法を指定する関連した portType の集まりを定義する。後の節でも述べるが、以下の事柄が言える。

- Factory portType を拡張する portType、もしくは NotificationSource portType のような新しいサービスを生成するよう定義された特化されたメソッドを持つ portType を考える。クライアントはこれらを実装するサービスインスタンスの createService 操作を呼び出すことで、グリッドサービスインスタンスの**生成**を要求することができる。
- クライアントは、サービスインスタンスに**明確な**破棄操作を要求するか、(GridService portType でサポートされる破棄操作（§9）を参照)、もしくはソフトステートアプローチによってグリッドサービスインスタンスの**破棄**を要求可能である。ソフトステートアプローチでは（[Grid Physiology] で動機付けと説明がなされたように)、クライアントがグリッドサービスインスタンスに対してある特定の期間関心があることを登録する。そのインスタンスが、タイムアウトを拡張するための関心の再確認をいずれのクライアントからも受信せずにその期間が過ぎた場合、そのサービスインスタンスを自動的に破棄してよい。定期的な再確認は、必要な限りグリッドサービスインスタンスの生存期間を拡張する働きを担うことができる（GridService portType の requestTerminationBefore/After 操作（§9）を参照)。

加えて、グリッドサービスインスタンスは§11 で定義される標準通知インターフェイスによって生存期間関連イベントの通知をサポート**可能である(MAY)**。

グリッドサービスインスタンスはソフトステート生存期間管理をサポートすることが**できる(MAY)**。その場合、クライアントはグリッドサービスインスタンスがファクトリ(§12)によって生成される際に初期サービスインスタンス生存期間を取り決める。さらに、権限を与えられたクライアントはそのサービスの生存期間の延長を要求するために requestTerminationBefore/After(「キープアライブ」)メッセージを送信することが**できる(MAY)**。そのグリッドサービスインスタンスの終了時刻に達したら、そのサービスインスタンスを受け持つサーバはそのサービスインスタンスを破棄し、関連付けられた資源を回収し、そのサービスインスタンスに関する全ての知識をその管理下のハンドルリゾルバから取り除いて**よい(MAY)**。

終了時刻は非単調に変化し**うる (MAY)**。すなわち、クライアントは現在の終了時刻よりも早い終了時刻を要求することが**できる(MAY)**。もし要求された終了時刻が現在時刻の前であれば、それは即座のもしくは優先的な破棄要求であると解釈され**なければならない (MUST)**。

グリッドサービスインスタンスは、任意時点で自身の生存期間を延長する決定を下すことが**できる(MAY)**。サービスインスタンスはまた、例えば資源制限や優先度のため資源を放棄するよう指示を受けた場合など、任意時点で自身を破棄することが**できる(MAY)**。

OGSI において終了時刻は以下の型を用いて表現される。

```
targetNamespace = "http://www.gridforum.org/namespaces/2003/03/OGSI"

<xsd:complexType name="TerminationTimeType">
   <xsd:attribute name="after" type="ogsi:ExtendedDateTimeType"
                  use="optional"/>
   <xsd:attribute name="before" type="ogsi:ExtendedDateTimeType"
                  use="optional"/>
   <xsd:attribute name="timestamp" type="xsd:dateTime"
                  use="optional"/>
</xsd:complexType>

<xsd:simpleType name="ExtendedDateTimeType">
   <xsd:union memberTypes="ogsi:InfinityType xsd:dateTime"/>
</xsd:simpleType>

<xsd:simpleType name="InfinityType">
   <xsd:restriction base="string">
      <xsd:enumeration value="infinity"/>
   </xsd:restriction>
</xsd:simpleType>
```

TerminationTimeType の *after* 属性は、サービスインスタンスの最も早い終了予定時刻を持つ。過去の値はサービスインスタンスが任意時刻に終了可能であることを示す。特別値 "infinity" はサービスインスタンスが無限に存在する予定であることを示す (ogsi:ExtendedDateTimeType は§7.2 で定義され、それは xsd:dateTime の値または特別値"infinity"を許す)。

TerminationTimeType の *before* 属性は、サービスインスタンスの最も遅い終了予定時刻を持つ。過去の値はサービスインスタンスが終了しつつあることを意味する。特別値 "infinity" はサービスインスタンスが終了する予定が無いことを意味する。*after* の時刻は *before* の時刻以前で**なければならない(MUST)**。

TerminationTimeType の *timestamp* 属性は、この TerminationTimeType が属するサービスインスタンスに知られているように、そのサービスインスタンスにおいて after 属性と before 属性が有効であると分かる時刻に等しい。この timestamp は、ある特定のサービスインスタンスに関連した TerminationTimeType 値を順序付けるために使用**可能である (MAY)**。さらに、ある状況においては、このサービスインスタンスの時刻とクライアントやなんらかの他の時刻とのずれを修正するために使用**可能である (MAY)**。

7.7 操作故障の共通処理

OGSI は、グリッドサービスが返さ**ねばならない(MUST)** 故障メッセージのための基底 XSD 型を定義する。これは、全ての故障メッセージが持つ情報の共通集合を持つことによって問題の判定を単純にする。

全ての OGSI 故障の基礎は ogsi:FaultType XSD 型である。

```
targetNamespace = "http://www.gridforum.org/namespaces/2003/03/OGSI"
```

```
<xsd:element name=”fault” type=”FaultType”>

<xsd:complexType name=“FaultType”>
   <xsd:sequence>
      <xsd:element name=“description”
                   type=“xsd:string”
                   minOccurs=“0” maxOccurs=“unbounded”/>
      <xsd:element name=“originator”
                   type=“ogsi:LocatorType”
                   minOccurs=“1” maxOccurs=“1”/>
      <xsd:element name=“timestamp”
                   type=“xsd:dateTime”
                   minOccurs=“1” maxOccurs=“1”/>
      <xsd:element name=“faultcause”
                   type=“ogsi:FaultType”
                   minOccurs=“0” maxOccurs=“unbounded”/>
      <xsd:element name=“faultcode”
                   type=“ogsi:FaultCodeType”
                   minOccurs=“0” maxOccurs=“1”/>
      <xsd:element name="extension"
                   type="ogsi:ExtensibilityType"
                   minOccurs="0" maxOccurs="1"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="FaultCodeType">
   <xsd:simpleContent>
      <xsd:extension base="xsd:string">
         <xsd:attribute name="faultscheme" type="anyURI"
                        use="required"/>
       </xsd:extension>
   </xsd:simpleContent>
</xsd:complexType>

<xsd:complexType name="ExtensibilityType">
   <xsd:sequence>
      <xsd:any namespace="##any"/>
   </xsd:sequence>
</xsd:complexType>
```

グリッドサービスインスタンスの全ての故障は、直接 FaultType を使用するか、FaultType を拡張して使用することが**推奨される(RECOMMENDED)**。だが、グリッドサービスインスタンスの故障が FaultType から派生しない状況が存在しても**よい (MAY)**。例えば、グリッドサービスが OGSI portType に加えレガシーportType を使用している場合などである。

任意の (OPTIONAL) *description* 要素は、その故障の自然言語による記述を持つ。この記述はユーザへの故障の説明の手助けとなることが期待される。description 要素は任意個数存在**可能である(MAY)**。

要求される (REQUIRED) *originator* 要素は故障発生源のサービスインスタンスへのロケータである。

要求される (REQUIRED) *timestamp* 要素は故障が発生した時刻である。

任意の(OPTIONAL) *faultcause* 要素は、結果としてこの故障を発生させた根本的な原因について記述する ogsi:FaultType 要素である。この要素は規約的に、FaultType を拡張したより特化した故障を記述する xsi:type と共に用いられる。faultcause 要素は任意個数存在**可能である(MAY)**。fault 要素中に faultcause 要素を含むことができることは、故障情報のサービス呼び出しスタックによる「連鎖」を可能にする。その結果、この故障メッセージの受信者が、その故障の理由の詳細についてより多く理解するためにその原因を突き詰めていくことが可能になる。

任意の (OPTIONAL) *faultcode* 要素は、レガシー故障報告システム (例えば POSIX errno)の簡便なサポートを提供する。faultcode の *faultscheme* 要素は、faultcode が解釈される**べき(SHOULD)**コンテキストを定義する URI (URL ではなく) であら**ねばならない (MUST)**。例えば、ある URI は POSIX errno の faultcode へのマッピング方法を説明するかもしれなく、その URI は POSIX errno を持つ任意の faultcode 要素に出現するだろう。

任意の(OPTIONAL) *extension* 要素は、存在するならば、他の XML 名前空間での故障に固有な追加情報を含む。*extension* 要素の中には任意個数の拡張性要素が存在**可能である(MAY)**。

グリッドサービス操作によって返される個々の故障は、WSDL 操作定義に別個の故障応答として記載され**なければならない(MUST)**。各操作の故障応答は fault 要素と同名を持た**ねばならない (MUST)**。かつ、その応答は "fault" と命名された 1 つの part 要素を持つ WSDL message 定義を参照せ**ねばばならない (MUST)**。ここで part 要素はその fault 要素を参照する "element=" 属性を持つ。全てのグリッドサービス操作は、操作固有の故障に加えて ogsi:fault を返さ**ねばならない(MUST)**。以下に例を挙げる。

```
<wsdl:definitions …>
   <types>
      <xsd:schema …>
         <xsd:complexType name="MyFaultType">
            <xsd:complexContent>
               <xsd:extension base="ogsi:FaultType/>
            </xsd:complexContent>
         </xsd:complexType>
         <xsd:element name="myFault" type="tns:MyFaultType"/>
      </xsd:schema>
   </types>

   <message name=myFaultMessage">
      <part name="fault" element="tns:myFault"/>
   </message>

   <gwsdl:portType …>
      <wsdl:operation …>
         <input …>
```

```
        <output …>
        <fault name="myFault" message="tns:myFaultMessage"/>
        <fault name="fault" message="ogsi:faultMessage"/>
      </wsdl:operation>
    </gwsdl:portType>
</wsdl:definitions>
```

グリッドサービス操作は、操作定義によって要求される特定の故障の代わりに、より改良された故障（すなわち XSD 拡張）を返しても**よい (MAY)**。例えば、ある操作がある状況下で myFault 要素を持つ故障を返すよう指定されているとする。このとき、その操作のある特定の実装は myFault 要素の代わりにその拡張を返しても**よい(MAY)**。これは改良された故障の xsi:type を持つ myFault メッセージを返すことによってなされる**べきである (SHOULD)**。

7.8 拡張可能操作

いくつかの OGSI 操作は型無し拡張性要素を入力変数として受理する。この要素は挙動の共通パターンを拡張可能な方法で表現することを可能にする。クライアントがそのうような操作でサポートされる有効な拡張を発見可能にするために、我々は静的サービスデータ値によって拡張可能操作ケイパビリティを表現する、共通のアプローチを定義する。

例えば NotificationSource::subscribe 操作は、クライアントがサービスインスタンスに、そのサービスの serviceDataValue の一部が変更されたときに常に通知メッセージを要求することを可能にする。通知を送信する対象のサービスデータの特定の部分は、登録式によって定義される。この引数は完全に型付けされているわけではなく、拡張可能な引数である。NotificationSource portType によって 1 つの単純な登録式 (subscription expression) が定義されており、それは NotificationSource を実装する任意のサービスにこの引数として渡すことが可能である。しかし、NotificationSource を継承する portType を実装するサービスは、より強力で特定の問題分野に特化した新たなクエリ式（query expression）を定義可能である。これによりこのサービスは登録のケイパビリティを拡張できる。本節では、NotificationSource portType の拡張を実装するサービスによってどの登録式がサポートされるかをクライアントが判別可能にする方法を定義する。

我々は、拡張可能な操作を記述する全ての SDE の基礎となる単一の XSD 型を定義する。

```
targetNamespace = “http://www.gridforum.org/namespaces/2003/03/OGSI”

<xsd:complexType name="OperationExtensibilityType">
  <xsd:attribute name="inputElement" type="QName" use=”optional”/>
</xsd:complexType>
```

portType の個々の拡張可能操作において、その portType は OperationExtensibilityType 型でかつ mutability=”static” である serviceData 宣言を持つ**べきである(SHOULD)**。この SDE の静的な値がこの操作の有効な拡張を定義する。

例えば、myOperation という名前の操作を持つ myPt という名前の portType を持つと仮定する。ここで、myOperataion 操作の引数の内 1 つが拡張可能であるとする。さらに、その myOperation の拡張可能入力引数に渡すことが可能な、myop:myOption1 と myop:myOption2 という名前の 2 つの標準入力要素があるとする。myOperation の portType 定義は以下のようになるだろう。

```
<gwsdl:portType name="myPT">
   …
   <sd:serviceData name="myOperationExtensiblity"
           type="ogsi:OperationExtensiblityType"
           minOccurs=0 maxOccurs="unbounded"
           mutability="static"
           modifiability="false"
           nillable="false" />
   …
   <sd:staticServiceDataValues>
      <myOperationExtensibility inputElement="m1:myOption1"/>
      <myOperationExtensibility inputElement="m1:myOption2"/>
   </sd:staticServiceDataValues>
   …
</gwsdl:portType>
```

この SDE の inputElement 属性は、その操作の特定の拡張の型と挙動をユニークに示す QName であら**ねばならない (MUST)**。inputElement は、存在するならば、拡張可能入力引数として操作に渡せる有効な要素の XSD 要素宣言の QName である**べきである (SHOULD)**。その拡張可能操作の全ての他のプロパティは、OperationExtensibilityType の拡張で明示的に定義されていない限りその inputElement によって示される。例えば、inputElement はその操作の出力引数の型を示すことや、その操作がこの inputElement を受け取るときの意味を示すことが**許されている (MAY)**。

inputElement が SDE 値から省略されるならば、操作を呼び出す際に拡張可能な入力引数を省略することが妥当で**なければならない(MUST)**。

操作拡張性 SDE 値は、操作拡張性 serviceData 宣言を持つ portType を拡張する portType に含まれれても**よい (MAY)**。例えば、myPT を拡張する myPT2 という別の portType は、以下のように myOperation に追加の有効な入力引数を定義することができる。

```
<gwsdl:portType name="myPT2" extends="m1:myPT>
   …
   <sd:staticServiceDataValues>
      <myOperationExtensibility inputElement="m2:myOption3"/>
      <myOperationExtensibility/>
   </sd:staticServiceDataValues>
   …
</gwsdl:portType>
```

この例において、myPT2 を実装するサービスインスタンスは myOperation の拡張可能な入力引数に対して 4 つのオプションをサポートする。すなわち m1:myOption1 と、

m1:myOption2、m2:myOption3、要素無しである。これらのオプションのそれぞれはさらに、出力引数の型、inputElement に関連する操作のセマンティクス、などを示すかもしれない。

ある状況においては、ogsi:OperationExtensibilityType を追加の属性や要素を含むよう拡張することが有用である。そのような場合、ogsi:OperationExtensibilityType の xsd:extension である新しい型が定義されるべきであり、この拡張された型を用いて serviceData 要素が定義されるべきである。例として、Factory portType (§12) を参照せよ。

8 グリッドサービスインターフェイス

本仕様は、WSDL portType と関連した挙動として、OGSI にとって重要と考えられる共通分散コンピューティングパターンを定義、体系化する。これらの OGSI portType は表 2 に挙げられており、以後の節で説明される。これらはすべて ogsi 名前空間において定義される。従って、OGSI 準拠コンポーネントの設計者の仕事は、GridService portType と、表 2 に挙げた他の任意の portType、またアプリケーションやドメイン固有の portType による組み合わせを拡張する portType を設計することである。

表 2: 本文書で定義される portType のまとめ

portType 名	節	説明
GridService	§9	サービスモデルのルート挙動をカプセル化する
HandleResolver	§10	GSH から GSR へのマッピング
NotificationSource	§11.1	クライアントが通知メッセージへ登録することを可能にする
NotificationSubscription	§11.2	単一の NotificationSource と NotificationSink ペアの関係を定義する
NotificationSink	§11.3	通知メッセージをサービスインスタンスへ届けるための単一の操作を定義する (サービスインスタンスはこの操作を実装する)
Factory	§12	グリッドサービスインスタンス作成のための標準操作
ServiceGroup	§13	クライアントがサービスグループを管理することを可能にする
ServiceGroupRegistration	§13.3	グリッドサービスが ServiceGroup に追加、かつそれから除去されることを可能にする
ServiceGroupEntry	§13.2	グリッドサービスインスタンスと ServiceGroup におけるそのメンバーシップとの関係を定義する

これらのすべての OGSI portType は ogsi 名前空間において定義されている。

9 GridService PortType

すべてのグリッドサービスは、GridService portType を実装**しなければならない (MUST)** (すなわち、グリッドサービス記述と関連付けられた portType のひとつでなければならない)。従って、GridService portType は OGSI における基本インターフェイス定義をなす。この portType は、コンポーネントモデルのルート挙動をカプセル化するという点で、Smalltalk や Java のようなオブジェクト指向プログラミング言語における Object クラスに類似している。グリッドサービスインスタンスの serviceData セットに対するクエリ、更新、そしてそのインスタンスの終了管理という挙動が、GridService portType によってカプセル化される。

Web サービスインターフェイスの設計において、文書中心メッセージングパターン (document-centric messaging patterns) と遠隔手続き呼び出し (RPC) のどちらかを選択しなければならない。これは、グリッドサービスインターフェイスの設計者も直面する選択である。GridService portType は型付き引数を用いたいくつかの操作を提供するが、それらの型付き引数のいくつかについては、多くの拡張性オプションが用意される。サービスデータは、特定のサービスインスタンスがどの extensibility 要素を受け付けるかを表すのに用いられる。グリッドサービス設計者は、portType をここで述べた機能を用いて構成し、その portType において文書中心アプローチと RPC アプローチを自由に混在させて構わない。

9.1 GridService: サービスデータ宣言

GridService portType は以下の serviceData 要素を含む。

- interface

  サービスインスタンスの完全なインターフェイス定義内の、すべての portType の QName。この集合は、このサービスインスタンスによって実装される portType の QName の推移閉包を含**まなければならない (MUST)**。

```
<sd:serviceData name="interface"
     type="xsd:QName"
     minOccurs="1" maxOccurs="unbounded"
     mutability="constant"
     modifiable="false"
     nillable="false"/>
```

- serviceDataName

  このサービスインスタンスでサポートされる個々のサービスデータ要素の QName。この集合は、この portType の WSDL 定義で宣言されるすべての service data 要素の QName を含**まなければならない (MUST)**。この集合はまた、このインスタンスによって動的に追加される service data 要素の QName を含み**うる (MAY)**。

```
<sd:serviceData name="serviceDataName"
     type="xsd:QName"
     minOccurs="0" maxOccurs="unbounded"
     mutability="mutable"
     modifiable="false"
```

```
      nillable="false"/>
```

- factoryLocator

  このグリッドサービスインスタンスを作成したファクトリのサービスロケータ。もしインスタンスがファクトリによって作成されたのではないならば、この値は xsi:nil で**なければならない (MUST)**。このロケータは、このグリッドサービスインスタンスの生存期間中、同じ 1 つのファクトリサービスを参照し**なければならない (MUST)**。しかし、このロケータのハンドルと参照はサービスインスタンスの生存期間中に変化**しうる (MAY)**。

```
<sd:serviceData name="factoryLocator"
     type="ogsi:LocatorType"
     minOccurs="1" maxOccurs="1"
     mutability="mutable"
     modifiable="false"
     nillable="true"/>
```

- gridServiceHandle

  このグリッドサービスインスタンスの 0 個以上のグリッドサービスハンドル。この集合に含まれない、このサービスインスタンスへのハンドルが存在しても**よい (MAY)**。

```
<sd:serviceData name="gridServiceHandle"
     type="ogsi:HandleType"
     minOccurs="0" maxOccurs="unbounded"
     mutability="extendable"
     modifiable="false"
     nillable="false"/>
```

- gridServiceReference

  このグリッドサービスインスタンスへの 1 個以上のグリッドサービス参照。1 つの service data value 要素は GSR の WSDL 表記で**なければならない (MUST)**。他の service data value 要素は、GSR の他の形式を表していてもよい。この集合に含まれない、このサービスインスタンスへの参照が存在しても**よい (MAY)**。

```
<sd:serviceData name="gridServiceReference"
     type="ogsi:ReferenceType"
     minOccurs="1" maxOccurs="unbounded"
     mutability="mutable"
     modifiable="false"
     nillable="false"/>
```

- findServiceDataExtensibility

  findServiceData 操作の操作拡張性宣言(§7.8) の集合。この SDE の値によって宣言された任意の正規の inputElement は、このインスタンスの

findServiceData 操作への QueryExpression 引数としてクライアントから使用**可能であり (MAY)**、このクエリの意味とその結果として返される値を示す。

```
<sd:serviceData name="findServiceDataExtensibility"
     type="ogsi:OperationExtensibilityType"
     minOccurs="1" maxOccurs="unbounded"
     mutability="static"
     modifiable="false"
     nillable="false"/>
```

- setServiceDataExtensibility

  setServiceData 操作の操作拡張性宣言(0) の集合。この SDE の値によって宣言された任意の正規の inputElement は、このインスタンスの setServiceData 操作への UpdateExpression 引数としてクライアントから使用**可能であり (MAY)**、更新の意味とその結果として返される値を示す。

```
<sd:serviceData name="setServiceDataExtensibility"
     type="ogsi:OperationExtensibilityType"
     minOccurs="2" maxOccurs="unbounded"
     mutability="static"
     modifiable="false"
     nillable="false"/>
```

- terminationTime

  このサービスインスタンスの終了時刻 (§7.6 を参照)。

```
<sd:serviceData name="terminationTime"
     type="ogsi:TerminationTimeType"
     minOccurs="1" maxOccurs="1"
     mutability="mutable"
     modifiable="false"
     nillable="false"/>
```

さらに、GridService portType は以下の service data value 要素の初期集合を定める。

```
<sd:staticServiceDataValues>

  <ogsi:findServiceDataExtensibility
       inputElement="ogsi:queryByServiceDataNames" />

  <ogsi:setServiceDataExtensibility
       inputElement="ogsi:setByServiceDataNames" />

  <ogsi:setServiceDataExtensibility>
       inputElement="ogsi:deleteByServiceDataNames" />

</sd:staticServiceDataValues>
```

9.2 GridService: 操作

9.2.1 GridService :: findServiceData

サービスデータを問い合わせる。

入力

- *QueryExpression:* 実行されるクエリ。この拡張可能な引数は、findServiceDataExtensibility SDE 値の 1 つで表された inputElement 宣言に従わ**なければならない (MUST)**。このサービスインスタンスは、この引数のルート要素のタグに基づいて行うべき動作を推定する。

出力

- *Result*: クエリの結果。結果の形式は QueryExpression による。

故障

- *ExtensibilityNotSupportedFault*: QueryExpression の型がこのサービスインスタンスではサポートされていないため、サービスインスタンスが QueryExpression を評価できない。
- *ExtensibilityTypeFault*: QueryExpression として渡された値が、その値の型に違反する。
- *TargetInvalidFault*: QueryExpression が要求する SDE の内、1 つまたは複数個がこのサービスインスタンス内に存在しない。
- *Fault*: 任意の他の故障

すべてのグリッドサービスインスタンスは、§9.2.1.1 で定義される queryByServiceDataNames に従う *QueryExpression* をサポートし**なければならない (MUST)**。グリッドサービスインスタンスは、他の *QueryExpression* をサポートしても**よい (MAY)**。

あるグリッドサービスインスタンスでサポートされるクエリ式 (query expression) の型のリストは、そのインスタンスの findServiceDataExtensibility SDE 値で表される。従って、クライアントはあるサービスインスタンスでサポートされるクエリ式の型を findServiceData 操作を用いて知ることができる。すなわち、そのインスタンスに対して、“ogsi:findServiceDataExtensibility” という name 子要素を持った queryByServiceDataNames 要素を用いて findServiceData 要求を実行すればよい。

クライアントがクエリ可能なサービスデータは、ポリシー制限により**うる (MAY)**。例えば、ある service data 要素はあるクライアントからは参照不可能になり**うる (MAY)**、かつ SDE のある service data value 要素はあるクライアントからは参照不可能となり**うる (MAY)**。

9.2.1.1 queryByServiceDataNames

queryByServiceDataNames の結果は、queryByServiceDataNames 引数で挙げられた service data 要素を含む serviceDataValues 要素である。queryByServiceDataNames 内

で挙げられた名前は、このサービスインスタンスに含まれる service data 要素（§9.1 を参照）に含まれ**なければならない（MUST）**。

queryByServiceDataNames 要素は以下のように定義される。

```
targetNamespace = “http://www.gridforum.org/namespaces/2003/03/OGSI”

<xsd:element name="queryByServiceDataNames" type="ogsi:QNamesType"/>

<xsd:complexType name="QNamesType">
   <xsd:sequence>
      <xsd:element name="name" type="QName"
                   minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
```

findServiceData 操作の queryByServiceDataNames クエリに対する *Result* 出力は、queryByServiceDataNames に挙げられた service data 要素を含む sd:serviceDataValues 要素で**なければならない（MUST）**。

例えば、以下の QueryExpression による findServiceData 呼び出しを考える。

```
<ogsi:queryByServiceDataNames>
   <name>ogsi:findServiceDataExtensibility</name>
   <name>ogsi:setServiceDataExtensibility</name>
</ogsi:queryByServiceDataNames>
```

この呼び出しは、以下の結果を返すだろう。

```
<sd:serviceDataValues>
   <ogsi:findServiceDataExtensibility
         inputElement=”ogsi:queryByServiceDataNames”/>
   <ogsi:setServiceDataExtensibility
         inputElement=”ogsi:setByServiceDataNames”/>
   <ogsi:setServiceDataExtensibility>
         inputElement=”ogsi:deleteByServiceDataNames”/>
</sd:serviceDataValues>
```

クライアントは、これらの返り値の妥当性を検証するために、serviceData と serviceDataValues の lifetime 属性（§7.3 を参照）を使うことが**できる（MAY）**。

9.2.2 GridService :: setServiceData

setServiceData 操作は、サービスデータ宣言が modifiable="true"としている場合に、service data 要素値の変更を可能にする。service data 要素を変更することは、そのサービスインスタンスの対応する状態を変更することを意味する。modifiable="true" である属性をもつ service data 要素がない場合は、setServiceData は本質的に無効化されていると言える。

入力

- *UpadateExpression*: 実行される更新。この拡張可能引数は、setServiceDataExtensibility SDE 値の 1 つで表された inputElement 宣言に従わ**なければならない (MUST)**。このサービスインスタンスは、この引数のルート要素のタグに基づいて行うべき動作を推定する。

出力

- *Result*: 更新の結果。この結果の形式は UpdateExpression による。

故障

- *ExtensibilityNotSupportedFault*: UpdateExpression の型がこのサービスインスタンスではサポートされていないため、サービスインスタンスが UpdateExpression を評価できない。
- *ExtensibilityTypeFault*: UpdateExpression として渡された値が、その値の型に違反する。
- *CarinalityVaiolationFault*: 要求された操作を実行すると、サービスの SDE の "minOccurs"、"maxOccurs" のどちらかまたは両方について違反する。
- *MutabilityVaiolationFault*: UpdateExparession が、このサービスの SDE の "mutability" 属性に矛盾した。
- *ModifiabilityVaiolationFault*: UpdateExpression が、このサービスの SDE の"modifiable" 属性に矛盾した。
- *TypeVaiolationFault*: UpdateExpression が、このサービスの SDE の XSD 型に従わない値を含む。
- *IncorrectValueFault*: UpdateExpression が、正しい XSD 型を持つが、他の理由のためにこのサービスインスタンスが受理不可能な値を含む。
- *PartialFailureFault*: このサービスインスタンスが、UpdateExpression を完全には満たすことができなかった。この故障は FaultType を拡張したものであり、更新不可であった UpdateExpression の SDE の qname リストを保持する要素を持つ。この故障は、1 個以上の "faultcause" 要素を持つことが**できる (MAY)**。この要素は、setServiceData で許されている任意の故障の型を用いて、失敗した部分をより詳細に説明するものである。
- *Fault*: 任意の他の故障

すべてのグリッドサービスインスタンスは、それぞれ§9.2.2.1 と §9.2.2.2 で定義される、setByServiceDataNames と deleteByServiceDataNames に従う *UpdateExpressions* を

サポートし**なければならない (MUST)**。グリッドサービスインスタンスは、他の *UpdateExpressions* をサポートしても**よい (MAY)**。

あるグリッドサービスインスタンスでサポートされる更新式 (update expression) の型のリストは、そのインスタンスの setServiceDataExtensibility SDE 値で表される。従って、クライアントはあるサービスインスタンスでサポートされる更新式の型を findServiceData 操作を用いて知ることができる。すなわち、そのインスタンスに対して、"ogsi:setServiceDataExtensibility" という name 子要素を持った queryByServiceDataNames 要素を用いて findServiceData 要求を実行すればよい。

クライアントが設定可能なサービスデータは、ポリシー制限により**うる (MAY)**。例えば、ある service data 要素はあるクライアントからは参照不可能**かもしれなく (MAY)**、かつ SDE のある service data value 要素はあるクライアントからは設定不可能となり**うる (MAY)**。

9.2.2.1 setByServiceDataNames

setByServiceDataNames は、同式内で挙げられた service data 要素を含む serviceDataValues 要素について、その要素内の SDE 値を更新する。同式内の名前は、modifiable="true"である属性を持った、このサービスインスタンスが保持する serviceDataNames (§9.1 を参照) に含まれ**なければならない (MUST)**。更新操作が特定の順序で行われる保証は一切ない。サービスインスタンスが、SDE 値のサービス状態についての更新の成功を保証する役目を持つ。サービスインスタンスは、可能な限り多くの SDE の更新を成功させ、失敗した SDE を返す**べきである (SHOULD)**。サービスインスタンスは、ある SDE の更新を失敗したとたんに終了し、継続動作しないことも**許されている (MAY)**。

部分的な失敗 (すなわち、ある SDE は更新されるが、別のある SDE は更新されない場合) は、更新不可であった SDE の QName のリストとともに、PartialFailureFault を返すことによって示され**なければならない (MUST)**。この故障は、式内の更新不可であった SDE それぞれについて 1 つの "faultcause" 要素を**持ってもよい (MAY)**。ここで、"faultcause" の型は有効な setServiceData 故障であり、かつその "faultcause" の extensibility 要素は更新不可であった setByServiceDataNames 式の SDE 値である。

setByServiceDataNames は、§6.2.3 で述べられたように SDE の mutability 属性に従わ**なければならない (MUST)**。同属性の値が "static" や "constant" である場合、setByServiceDataNames の実行は許されない。"extensible" である場合、setByServiceDataNames は新しい要素を SDE の既存の値に付加し**なければならない (MUST)**。"mutable" ならば、setByServiceDataNames は既存の SDE 値を与えられた値で置き換え**なければならない (MUST)**。setByServiceDataNames の結果は、式内で挙げられた SDE それぞれについて、serviceData 宣言で定義された minOccurs/maxOccurs ルールに従う serviceData 要素値とな**らなければならない (MUST)**。

この型の非正規な文法は以下の通りである。

```
<ogsi:setByServiceDataNames>
   <someServiceDataNameTag>
      new SDE value(s)
   </someServiceDataNameTag>*
   <someOtherServiceDataNameTag>
```

```
      new SDE value(s)
   </someOtherServiceDataNameTag>*
</ogsi:setByServiceDataNames>
```

setServiceData 操作の setByServiceDataNames 呼び出しに対する *Result* 出力は、setByServiceDataNames 式で挙げられた、値の置き換えに失敗した service data 要素を含む serviceDataValues 要素で**なければならない (MUST)**。空の値の集合は成功したことを意味する。

クライアントは、サービスの serviceDataValues の値が互いに一貫したものであることを仮定する**べきでない (SHOULD NOT)** ことに注意せよ。クライアントはまた、service data 要素の置き換えが一貫性が保たれた状態で行われることを仮定する**べきでない (SHOULD NOT)**。

9.2.2.2 deleteByServiceDataNames

deleteByServiceDataNames は、同式内で挙げられた service data 要素のすべての SDE 値を削除する。同式内の名前は、modifiable="true"である属性を持った、このサービスインスタンスが保持する serviceDataNames (§9.1 を参照) に含まれ**なければならない (MUST)**。削除が特定の順序で行われる保証は一切ない。サービスインスタンスが、SDE 値のサービス状態についての削除の成功を保証する役目を持つ。サービスインスタンスは、可能な限り多くの SDE の削除を成功させ、失敗した SDE を返す**べきである (SHOULD)**。サービスインスタンスは、ある SDE の削除を失敗したとたんに終了し、継続動作しないことも**許されている (MAY)**。

部分的な失敗 (すなわち、ある SDE は削除されるが、別のある SDE は削除されない場合) は、削除不可であった SDE の QName のリストとともに、PartialFailureFault を返すことによって示され**なければならない (MUST)**。この故障は、式内の更新不可であった SDE それぞれについて 1 つの "faultcause" 要素を持っても**よい (MAY)**。ここで、"faultcause" の型は有効な deleteServiceData 故障であり、かつその "faultcause"の extensibility 要素は更新不可であった QName を持つ。

deleteByServiceDataNames は、§6.2.3 で述べられたように SDE の mutability 属性に従わ**なければならない (MUST)**。同属性の値が "static" や "constant" 、"extendable" である場合、deleteByServiceDataNames の実行は許されない。"mutable" ならば、deleteByServiceDataNames はその SDE 名である要素をすべて削除する。deleteByServiceDataNames の結果は、式内で挙げられた SDE それぞれについて、serviceData 宣言で定義された minOccurs/maxOccurs ルールに従う serviceData 要素値とな**らなければならない (MUST)**。

deleteByServiceDataNames 要素は以下の通りに定義される。

```
targetNamespace = "http://www.gridforum.org/namespaces/2003/03/OGSI"

<xsd:element name="deleteByServiceDataNames" type="ogsi:QNamesType"/>
```

ogsi:QNamesType は 9.2.1.1 節で定義される。

setServiceData 操作の deleteByServiceDataNames 呼び出しに対する *Result* 出力は、deleteByServiceDataNames 式で挙げられた、削除に失敗した service data 要素を含む serviceDataValues 要素で**なければならない (MUST)**。空の値の集合は成功したことを意味する。

クライアントは、サービスの serviceDataValues の値が更新される前に互いに一貫したものであることを仮定する**べきでない (SHOULD NOT)** ことに注意せよ。クライアントはまた、service data 要素の削除が一貫性が保たれた状態で行われることを仮定する**べきでない (SHOULD NOT)**。

9.2.3 GridService :: requestTerminationAfter

requestTerminationAfter 操作は、サービスインスタンスの終了時刻の変更を要求する。この要求は、望ましい最も早い終了時刻を指定する。サービスインスタンスはこの要求を受けて、自身のポリシーと要求された時間に基づいて必要ならば終了時刻を調節して**よい (MAY)**。クライアントは応答を受けると、その応答の CurrentTimestamp に基づいて、順序通り到着しなかった応答を破棄**すべきである (SHOULD)**。

入力:

- *TerminationTime*: クライアントにとって許容可能な、グリッドサービスインスタンスの最も早い終了時刻。その型は、§7.2 で定義される ExtendedDataTimeType である。時刻が過去の場合は、クライアントは終了時刻に関心がないことを意味する。"infinity" という特別な値は、クライアントがサービスインスタンスに無限に継続して動作することを要求することを意味する。

出力::

- *CurrentTerminationTime*: サービスの現在の予定されている終了時刻を表す、TerminationTimeType 型（§7.6 で定義）の要素。CurrentTerminationTime の timestamp 属性は、サービスインスタンスがこの要求を処理した時刻で**なければならない (MUST)**。

故障:

- *TerminationTimeUnchangedFault*: サービスインスタンスは要求された終了時刻を無視した。
- *Fault*: 任意の他の故障。

9.2.4 GridService :: requestTerminationBefore

requestTerminationBefore は、サービスインスタンスの終了時刻の変更を要求する。この要求は、望ましい最も遅い終了時刻を指定する。サービスインスタンスはこの要求を受けて、自身のポリシーと要求された時間に基づいて必要ならば終了時刻を調節して**よい (MAY)**。クライアントは応答を受けると、その応答の timestamp に基づいて、順序通り到着しなかった応答を破棄**すべきである (SHOULD)**。

入力:

- *TerminationTime*: クライアントにとって許容可能な、グリッドサービスインスタンスの最も遅い終了時刻。その型は、§7.2 で定義される

ExtendedDataTimeType である。時刻が過去の場合は、クライアントはサービスインスタンスが可能な限り早く終了することを望んでいることを意味する。"infinity" という特別な値は、クライアントがサービスインスタンスの終了時刻に関心がないことを意味する。

出力:

- *CurrentTerminationTime*: サービスの現在の予定されている終了時刻を表す、TerminationTimeType 型（§7.6 で定義）の要素。CurrentTerminationTime の timestamp 属性は、サービスインスタンスがこの要求を処理した時刻で**なければならない (MUST)**。

故障:

- *TerminationTimeUnchangedFault*: サービスインスタンスは要求された終了時刻を無視することを選択した。
- *Fault*: 任意の他の故障。

9.2.5 GridService :: destroy

destroy 操作は、サービスインスタンスの破棄を明示的に要求する。グリッドサービスインスタンスは明示的な破棄要求を受けて、(1) 自身の破棄処理を開始し、破棄要求メッセージを受理したことを知らせる応答メッセージを返す、または、(2) 要求を無視し、失敗を意味する故障メッセージを返すのどちらかの処理を行わ**なければならない (MUST)**。グリッドサービスインスタンスの破棄処理が開始されたならば、そのサービスインスタンスは以降の要求に応答する**べきでない (SHOULD NOT)**。destroy 操作が成功すると、クライアントはそのサービスインスタンスの存在を仮定しては**いけない (MUST NOT)**。

入力:

- 無し (空の入力メッセージ)。

出力:

- 無し (destroy 操作が開始されたことを知らせる、空の出力メッセージ)。

故障:

- *ServiceNotDestroyedFault*: サービスインスタンスが自身の破棄処理を開始しなかった。
- *Fault*: 任意の他の故障。

10 HandleResolver PortType

HandleResolver portType は、GSH の特定の URI スキームによらない、GSH を GSR へ解決する標準手段を定義する。HandleResolver portType を実装するサービスインスタンスは**ハンドルリゾルバ**と呼ばれる。

ハンドルリゾルバは、GSH から GSR へのマッピングの持ち方に関して、異なるアプローチを持つことが許されている。ハンドルリゾルバは、ホスティング環境の生存期間管理サービスに直接関連付けられ**うる (MAY)**。この場合は、あるホスティング環境固有の特

殊な方法によって、インスタンスの作成と破棄が自動的にマッピングの追加、除去を行う。またハンドルリゾルバサービスは ServiceGroup portType を実装しても**よい (MAY)**。この場合は、サービスインスタンスが自身の存在をそのリゾルバに登録すると、リゾルバがマッピングデータベースを構成するために、そのインスタンスの service data 要素である gridServiceHandle と gridServiceReference に問い合わせを行う。ユーザ定義の portType によるユーザ定義の登録プロトコルを実装するハンドルリゾルバサービスも許されている。しかしこれらのいずれの場合においても、HandleResolver portType は GSH から GSR へのマッピングをリゾルバサービスに問い合わせるために使うことが**できる (MAY)**。

この portType は GridService portType を拡張する。

10.1 HandleResolver: サービスデータ宣言

HandleResolver portType は以下の serviceData 要素を含む。

- **handleResolverScheme**

  このハンドルリゾルバサービスインスタンスが GSR へ解決可能であって**よい (MAY)**、GSH スキームを表す URI の集合。handleResolverScheme URI の内、関係のある部分はその URI のスキーム部のみである（すなわち、":"の前に位置する部分)。例えば、"abc:def" という handleResolverScheme URI は、このハンドルリゾルバが "abc" というスキームの GSH を解決可能であって**よい (MAY)** ことを意味する。xsi:nil handleResolverScheme 値は、このリゾルバが任意スキームの任意 GSH を解決可能であって**よい (MAY)** ことを意味する。

```
<sd:serviceData name="handleResolverScheme"
     type="xsd:anyURI"
     minOccurs="1" maxOccurs="unbounded"
     mutability="mutable"
     modifiable="false"
     nillable="true"/>
```

10.2 HandleResolver: 操作

10.2.1 HandleResolver :: findByHandle

findByHandle 操作は、HandleSet ロケータのグリッドサービスハンドルに対する 1 つ以上のグリッドサービス参照を含むロケータを返す。

Input

- *HandleSet*: 1 つ以上のグリッドサービスハンドルを持つロケータ。ここで、(ロケータの定義上) ハンドルはすべて必ず同じグリッドサービスインスタンスを参照する。リゾルバは、この HandleSet の任意の 1 つ又は複数の GSH に基づいて解決して**よい (MAY)**。HandleSet は任意個数の GSR を持ち**うる (MAY)**。この GSR をハンドルリゾルバは無視するか、もしくはその GSH の以前の有効な解決とみなして**よい (MAY)**。
- *GSRExclusionSet*: (任意) 1 つ又は複数の GSR を保持するロケータ。それらの GSR は、クライアントがすでに保持しているが、ある理由のために不十分なもの

である。これらの GSR は、出力 Locator で返されるべきではない (SHOULD NOT)。GSRExclusionSet は任意個数の GSH を持ちうるが (MAY)、リゾルバは GSH の解決する際にそれらを無視しなければならない (MUST)。

出力

- *Locator*: 入力 HandleSet に対する 1 つ又は複数の GSR を持つサービスロケータ。このロケータはまた、入力 HandleSet 内のすべての GSH を含まなければならなく (MUST)、同グリッドサービスインスタンスへの追加 GSH を含んでもよい (MAY)。

故障

- *InvalidHandleFault*: ハンドルが URI スキームのシンタックスに違反している。
- *NoAdditionalReferencesAvailableFault:* リゾルバが、GSRExclusionSet 入力引数にまだ含まれていない GSR を返すことができない。
- *NoReferencesAvailableFault:* リゾルバが、GSRExclusionSet 入力引数によらず、入力ハンドルに対する GSR を返すことができない。以下の故障はこの NoReferecesAvailale 故障を拡張する。
    - *NoSuchServiceFault*: このハンドルに対するサービスインスタンスは存在しなかった、もしくはすでに終了した。この故障はあるいくつかの URI スキームに対してのみ適用可能であってよい (MAY)。
    - *NoSuchServiceStartedFault*: このハンドルに対するサービスインスタンスは存在しなかった。この故障はあるいくつかの URI スキームに対してのみ適用可能であってよい (MAY)。
    - *ServiceHasTerminatedFault*: このハンドルに対するサービスインスタンスは終了した。この故障はあるいくつかの URI スキームに対してのみ適用可能であってよい (MAY)。
    - *TemporarilyUnavailableFault*: このハンドルは有効なサービスインスタンスを参照するが、この時点では有効な参照へ解決できない。ただし、後に解決可能であってよい (MAY)。この故障はサービスインスタンスが参照可能になるかもしれない時刻を任意で返す。この故障はあるいくつかの URI スキームに対してのみ適用可能であってよい (MAY)。
- *RedirectionFault*: クライアントが要求可能 (MAY) な、代替ハンドルリゾルバが存在する。この故障は FaultType を拡張し、代替ハンドルリゾルバのロケータを含む要素を追加する。
- *Fault*: 任意の他の故障。

11 通知

通知の目的は、以下で説明するように、通知ソース (notification source) から通知シンク (notification sink) へと関心のあるメッセージを届けることである。

- **通知ソース**とは、NotificationSource portType を実装するグリッドサービスインスタンスである。ソースは任意個数のシンクへ通知メッセージを送信可能であって**よい (MAY)**。
- **通知シンク**とは、任意個数のソースから通知メッセージを受信するグリッドサービスインスタンスである。シンクは、NotificationSink portType を実装し**なければならなく (MUST)**、これによりシンクは通知メッセージを受信できるようになる。
- **通知メッセージ**とは、通知ソースから通知シンクへと送られる XML 要素である。同要素の型は、以下の登録式によって決定される。
- **登録式** (subscription expression) とは、通知ソースから通知シンクへどのようなメッセージが送られるべきかを表す XML 要素である。登録式はまた、サービスインスタンスの serviceDataValues の値の変更に基づいて、メッセージが届けられるべき時を表す。
- どのような通知メッセージがどこへ届けられるべきかについての合意を確立するために、**登録**要求がソースに対して発行される。これは、登録式、通知メッセージが送られるべき通知シンクのロケータ、この登録の初期有効期間を含む。
- 登録要求は、NotificationSubscription portType を実装するグリッドサービスインスタンスを作成する。このインスタンスは *subscription* と呼ばれる。クライアントは、この portType を登録の（ソフトステート）有効期間を管理するためと登録のプロパティを発見するために用いて**よい (MAY)**。

この通知フレームワークは、サービスからサービスへの通知メッセージの配送と、さまざまな仲介配送サービスの統合を可能にする。仲介配送サービスは、メッセージサービス、メッセージフィルタリングサービス、メッセージアーカイブ・再送サービスを含むかもしれない。

登録をグリッドサービスインスタンスとして扱うことは、登録を他のグリッドサービスと同じインターフェイスを用いて管理することを可能にする。OGSI の実装では、他のグリッドサービスインスタンスより軽量な、登録のための特化された実装技術を使うことが期待されるだろう。

11.1 NotificationSource PortType

NotificationSource portType は、クライアントが、この portType を実装するグリッドサービスインスタンスからの通知メッセージに登録することを可能にする。
NotificationSource portType は GridService portType を拡張する。

NotificationSource portType は GridService portType を拡張する。

11.1.1 NotificationSource: サービスデータ宣言

NotificationSource portType は以下の serviceData 要素を持つ。

- notifiableServiceDataName

  要求者が変更通知について登録**可能 (MAY)** な service data 要素の QName の集合。

```
<sd:serviceData name="notifiableServiceDataName"
     type="xsd:QName"
     minOccurs="0" maxOccurs="unbounded"
     mutability="mutable"
     modifiable="false"
     nillable="false"/>
```

- subscribeExtensibility

  登録操作の操作拡張性宣言（§7.8）の集合。クライアントは、この SDE の値によって宣言された任意の正規 inputElement を、このインスタンスの登録操作の SubscriptionExpression 引数として使用**可能 (MAY)** である。また同 inputElement は、登録の意味とこの登録対象である通知メッセージについての情報を提供する。

```
<sd:serviceData name="subscribeExtensibility"
     type="ogsi:OperationExtensibilityType"
     minOccurs="1" maxOccurs="unbounded"
     mutability="static"
     modifiable="false"
     nillable="false"/>
```

また、NotificationSource portType は以下の初期 service data value 要素を含む。

```
<sd:staticServiceDataValues>
   <ogsi:subscribeExtensibility
        inputElement="ogsi:subscribeByServiceDataNames" />
</sd:staticServiceDataValues>
```

11.1.2 NotificationSource: 操作

11.1.2.1 NotificationSource :: subscribe

対象インスタンスのサービスデータの、以降の変更の通知に登録する。この操作は、グリッドサービスインスタンス subscription を作成する。このインスタンスは以後、この登録の有効期間の管理とプロパティの発見のために使用**可能である (MAY)**。

入力:

- *SubscriptionExpression*: 実行される登録。この拡張可能な引数は、subscribeExtensibility SDE 値の 1 つで表された inputElement 宣言に従わ**なければならない (MUST)**。このサービスインスタンスは、この引数のルート要素のタグに基づいて行うべき動作を推定する。
- *Sink*: メッセージが配送される通知シンクのロケータ。これは、登録要求を発行するサービスインスタンスではない他のインスタンスへのロケータであって**よい (MAY)**。従って、第三者の登録が可能である。このロケータは、参照のみを持つことが**許されている (MAY)**。ゆえに、例えば NotificationSink portType を実装するがハンドルを持たない Web サービスを参照することができる。

- *ExpirationTime*: この subscription インスタンスが終了するべき初期時刻。インスタンスが終了すると、このシンクへの通知配送は停止する。subscription インスタンスの生存期間を変更するために、通常の GridService 生存期間管理操作を使うことが**できる (MAY)**。

出力:

- *SubscriptionInstanceLocator*: この登録を管理するために作成された subscription インスタンスへのロケータ。この subscription インスタンスは NotificationSubscription portType を実装し**なければならない (MUST)**。
- *CurrentTerminationTime*: (§7.6 で定義される)TerminationTimeType 型の要素。NotificationSubscription サービスの現在の予定終了時刻を表す。この CurrentTerminationTime の timestamp 属性は、NotificationSubscription サービスインスタンスが作成された時刻で**なければならない (MUST)**。

故障:

- *ExtensibilityNotSupportedFault*: SubscriptionExpression の型がこのサービスインスタンスではサポートされていないため、インスタンスがその SubscriptionExpression を評価できない。
- *ExtensibilityTypeFault*: SubscriptionExpression として渡された値が、その型に違反する。
- *TargetInvalidFault*: SubscriptionExpression が要求する 1 つ以上の SDE がこのサービスインスタンス中に存在しない。
- *Fault*: 任意の他の故障。

NotificationSource portType を実装するすべてのグリッドサービスインスタンスは、§11.1.2.1.1 で定義される subscribeByServiceDataNames の、subscribeExtensibility SDE 値をサポートし**なければならない (MUST)**。グリッドサービスインスタンスは、他の subscribeExtensibility SDE 値をサポートしても**よい (MAY)**。

あるグリッドサービスインスタンスでサポートされる登録式の型のリストは、そのインスタンスの subscribeExtensibility SDE 値として表される。従って、クライアントはインスタンスに対して findServiceData 要求を行うことで、そのインスタンスでサポートされる登録式を発見できる。すなわち、そのインスタンスに対して、“ogsi:subscribeExtensibility” という name 子要素を持った queryByServiceDataNames 要素を用いて findServiceData 要求を実行すればよい。

11.1.2.1.1 subscribeByServiceDataNames

subscribeByServiceDataNames によって、指定した service data 要素に変更があったときに常に通知メッセージが送られることになる。

subscribeByServiceDataNames 要素は以下のように定義される。

```
targetNamespace = “http://www.gridforum.org/namespaces/2003/03/OGSI”

<xsd:element name="subscribeByServiceDataNames"
         type="ogsi:SubscribeByNameType"/>
```

```
<xsd:complexType name="SubscribeByNameType">
   <xsd:complexContent>
      <xsd:extension base="ogsi:QNamesType">
         <xsd:attribute name="minInterval"
                        type="duration"
                        use="optional"/>
         <xsd:attribute name="maxInterval"
                        type="ogsi:MaxIntervalType"/>
      </xsd :extension>
   </xsd:complexContent>
</xsd:complexType>

<xsd:simpleType name="MaxIntervalType">
   <xsd:union memberTypes="ogsi:InfinityType xsd:duration"/>
</xsd :simpleType>
```

ogsi:QNamesType は§9.2.1.1 で定義される。ogsi:InfinityType は§7.6 で定義される。

minInterval プロパティは、通知メッセージの最小間隔を xsd:duration として指定する。このプロパティが指定されない場合は、通知ソースが間隔を選択して**よい (MAY)**。通知ソースはまた、要求された最小間隔を満たせない場合、登録要求を拒否しても**よい (MAY)**。

maxInterval プロパティは、通知メッセージの最大間隔を xsd:duration として指定する。この間隔の経過後に、指定された service data 要素の値が変化しなかったならば、ソースは同じ値を再送し**なければならない (MUST)**。この値が "infinity" のときは、ソースはサービスデータ値に変更がない限り、再送する必要はない。このプロパティが指定されなかった場合は、通知ソースが値を選択して**よい (MAY)**。

subscribeByServiceDataNames 登録では、通知ソースから通知シンクへと送られる通知メッセージの型は、要求された serviceDataName それぞれに対応するすべての SDE の値を保持する serviceDataValues 要素で**なければならない (MUST)**。これは、たとえ最後のメッセージからある一部の要素にのみ変更があった場合にもあてはまる。

11.2 NotificationSubscription PortType

通知に登録すると、*subscription* と呼ばれるグリッドサービスインスタンスが作成される。同インスタンスは NotificationSubscription portType を実装し**なければならない (MUST)**。NotificationSubscription portType は GridService portType を拡張する。クライアントは、このインスタンスを、登録の有効期間の管理とプロパティの発見のために使用**可能である (MAY)**。

NotificationSubscription portType は GridService portType を拡張する。

11.2.1 NotificationSubscription: サービスデータ宣言

NotificationSubscription portType は以下の serviceData 要素を含む。

- subscriptionExpression

この subscription インスタンスが管理する、現在の登録式。

```
<sd:serviceData name="subscriptionExpression"
     type="xsd:anyType"
     minOccurs="1" maxOccurs="1"
     mutability="mutable"
     modifiable="false"
     nillable="false"/>
```

- sinkLocator

  この subscription がメッセージを配送する通知シンクへのグリッドサービスロケータ。

```
<sd:serviceData name="sinkLocator"
     type="ogsi:LocatorType"
     minOccurs="1" maxOccurs="1"
     mutability="mutable"
     modifiable="false"
     nillable="false"/>
```

11.2.2 NotificationSubscription: 操作

無し。

11.3 NotificationSink PortType

NotificationSink portType は、通知メッセージをサービスインスタンスへ配送するための単一の操作を定義する。ここで、配送先のサービスインスタンスはこの操作を実装するインスタンスである。

注意: 本仕様で述べられた他のすべて portType とは異なり、NotificationSink portType を実装する Web サービスは、グリッドサービスインスタンスである必要はない。すなわち、NotificationSink portType を実装する Web サービスは GridService portType を実装する必要はない。

11.3.1 NotificationSink: サービスデータ宣言

無し。

11.3.2 NotificationSink: 操作

11.3.2.1 NotificationSink :: deliverNotification

このサービスへメッセージを配送する。

入力:

- *Message*: 通知メッセージを持つ XML 要素。メッセージの中身は通知登録に依存する。

この操作は入力のみを持ち、出力や故障を返すことはない。

12 Factory PortType

ファクトリとは抽象的な概念やパターンのことである。ファクトリは、クライアントがグリッドサービスインスタンスを作成するために使うグリッドサービスインスタンスに対応する。クライアントはファクトリに対して、作成操作を呼び出し、新規作成されたサービスインスタンスに対するロケータを受け取る。ファクトリは、文書中心 (document-centric) Factory portType を実装するグリッドサービスインスタンス、またはより特化されたファクトリ操作 (例えば、§11.1.2.1 で説明された NotificationSource::subscribe 操作) を実装するグリッドサービスインスタンスであって**よい (MAY)**。

ファクトリによる作成を受けて、新規作成されたグリッドサービスインスタンスは、ハンドル解決サービス (§•を参照) へ登録し、同サービスから GSH を受け取る**べきである (SHOULD)**。この登録を行うメソッドはホスティング環境に固有なものであり、本仕様の範囲外である。

Factory portType は GridService portType を拡張し**なければならない (MUST)**。

12.1 Factory: サービスデータ宣言

Factory portType は以下の serviceData 要素を含む。

- createServiceExtensibility

  createService 操作の操作拡張性宣言 (§7.8) の集合。クライアントは、この SDE の値によって宣言された任意の正規 inputElement を、このインスタンスの createService 操作の CreationParameters 引数として使用**可能 である (MAY)**。また同 inputElement は、作成の意味とこの作成による返り値についての情報を提供する。

```
<sd:serviceData name="createServiceExtensibility"
     type="ogsi:CreateServiceExtensibilityType"
     minOccurs="1" maxOccurs="unbounded"
     mutability="static"
     modifiable="false"
     nillable="false"/>
```

  ogsi:CreateServiceExtensibilityType は、ogsi:OperationExtensibilityType を拡張し、この inputElement の結果として作成されたサービスインスタンスが実装する、portType の集合を含む要素を定義する。この型は以下のように定義される。

```
<xsd:complexType name="CreateServiceExtensibilityType">
   <xsd:complexContent>
      <xsd:extension base="ogsi:OperationExtensibilityType">
         <xsd:sequence>
            <xsd:element name="createsInterface" type="xsd:QName"
                     minOccurs="1" maxOccurs="unbounded"/>
         </xsd:sequence>
      </xsd:extension>
```

```
    </xsd:complexContent>
  </xsd:complexType>
```

12.2 Factory: 操作

12.2.1 Factory :: createService

ファクトリの createService 操作は新たにグリッドサービスインスタンスを作成する。ソフトステート生存期間管理（§7.6）をサポートするために、クライアントは許容可能な自身の最も早い初期終了時刻と最も遅い初期終了時刻の期間を指定してもよい。ファクトリはこの期間の中で初期終了時刻を選択し、その時刻を作成要求の応答メッセージの一部としてクライアントへ返す。ファクトリはまた、クライアントが以降この新規作成されたグリッドサービスインスタンスに対して要求可能な、最長生存期間拡張を返す。そうではなく、要求された終了時刻がファクトリにとって受理不可能である場合は、グリッドサービス作成要求は失敗してもよい。

入力

- *TerminationTime* (任意): このクライアントが許容可能な、グリッドサービスインスタンスの最も早い初期終了時刻と最も遅い初期終了時刻。これらの値は§7.6で定義された terminationTime 要素として表される。
- *CreationParameters* (任意): サービスインスタンスの作成に必要な情報を持つ、ファクトリに固有な要素。この拡張可能な引数は、createServiceExtensibility SDE 値の 1 つで表された inputElement 宣言に従わ**なければならない (MUST)**。注意: このサービスインスタンスは、この引数のルート要素のタグに基づいて行うべき動作を推定する。

出力

- *Locator*: 新たに作成されたグリッドサービスインスタンスへのロケータ (§7.5.3 を参照)。
- *CurrentTerminationTime*: 新規作成されたサービスの現在予定されている終了時刻を表す、TerminationTimeType 型（§7.6 で定義）の要素。CurrentTerminationTime の timestamp 属性は、新規サービスインスタンスが作成された時刻で**なければならない (MUST)**。
- *ExtensibilityOutput* (任意): このファクトリと同ファクトリが作成するサービスに固有な XML 拡張性要素。

故障:

- *ExtensibilityNotSupportedFault:* CreationParameters の型がこのサービスインスタンスではサポートされていないため、インスタンスがその CreationParameters を評価できない。
- *ExtensibilityTypeFault*: CreattionParameters として渡された値が、その型に違反する。

- *ServiceAlreadyExistsFault*: 要求されたサービスインスタンスがすでに存在する。この故障は FaultType を拡張し、すでに存在するサービスインスタンスへのロケータである要素を追加する。
- *Fault*: 任意の他の故障。

13 ServiceGroup

ServiceGroup は他のグリッドサービスのグループに関する情報を管理するグリッドサービスインスタンスである。これらのサービスは、例えば連合サービスの一部であるといった特定の理由によりグループの一員となっているかもしれない。もしくはそれらは、例えば検索のための索引やレジストリに含まれるサービスといった、特定の関係を持たないサービスかもしれない。伝統的なグリッドサービスレジストリは、ServiceGroup で記述される基本挙動を拡張する portType によって定義可能だろう。

3 つの portType がサービスグループのインターフェイスを提供する。すなわち、ServiceGroup、ServiceGroupEntry、ServiceGroupRegistration である。

13.1 *ServiceGroup portType*

ServiceGroup portType は、0 個以上の**メンバー**サービスからなる**サービスグループ**を表現するためのインターフェイスを提供する。ServiceGroup の Entry SDE はグループ内の各メンバーグリッドサービスインスタンスのための要素を持つ。Entry SDE の ServiceGroupEntryLocator 要素は、ServiceGroupEntry portType (§13.2 参照) を実装するサービスインスタンスを参照する**べきである (SHOULD)**。同インスタンスはそのエントリのための管理機能を提供する。特に、それは個々のエントリに対する独立した生存期間管理機能とそのエントリへのユニークなキー (GSH) を提供する。さらに、より高度なエントリ管理機能を提供するために拡張可能である。

ServiceGroup portType は GridService portType を拡張する。

以下のプロパティは ServiceGroup、ServiceGroupEntry、ServiceGroupRegistration、ServiceGroup のメンバーグリッドサービスインスタンスに適用される。

- グリッドサービスインスタンスは、いくつかの ServiceGroup のメンバーになることが**可能である (MAY)**。
- ServiceGroup のメンバーグリッドサービスインスタンスは、異なる portType を実装**できる (MAY)**。
- ServiceGroupEntry は、それ自身の生存期間管理によって ServiceGroup から除去され**うる (MAY)**。
- ひとたび ServiceGroup が破棄されると、クライアントは ServiceGroupEntry サービスの破棄の責任を負わない。
- ひとたび ServiceGroup が破棄されると、クライアントは ServiceGroupEntry サービスの存在やその内容 (例えば生存期間プロパティ) の有効性に関して、一切の仮定を置けない。
- ServiceGroupEntry は高々1 つの ServiceGroup に所属し**なければならない (MUST)**。

- もし ServiceGroup に含まれる GridService が終了しても、ServiceGroup はこれを反映する必要は無い。
- ServiceGroup の全てのメンバーグリッドサービスインスタンスは、ServiceGroup の membershipContentRule SDE に列挙される portType の少なくとも 1 つに従わ**なければならない (MUST)** (すなわち同じ portType か、そのサブタイプである)。
- 1 つのメンバーグリッドサービスインスタンスが、複数回 1 つの ServiceGroup に含まれることが**許されている (MAY)**。ServiceGroup は entry の「かばん」(“bag”)である。ServiceGroup を拡張する portType の設計者は、そのセマンティクスを「集合」(“set”) や他の集まりの形に拡張することが**できる(MAY)**。

13.1.1 ServiceGroup: サービスデータ宣言

- membershipContentRule

  この SDE は、1 つの portType (memberInterface)と XSD 要素 QName (content)の集合を結びつける構造体を含む。ServiceGroup のメンバーである個々のサービスインスタンスは、ServiceGroup の membershipContentRule SDE 値に列挙される memberInterface の内、1 つ以上を実装し**なければならない (MUST)**。
  ServiceGroup の entry は、その entry のメンバーサービスが実装する任意の memberInterface と関連付けられた、全ての content 要素を含ま**なければならない(MUST)**。

  この SDE は、その ServiceGroup の entry SDE 値と、その ServiceGroup 内のサービスに対する ServiceGroupEntry の content SDE 値に対する、「データ型不変式」として機能する。すなわち、メンバーシップが membershipContentRule SDE 値で指定される制限を満たす、もしくは従うグリッドサービスに制約される。もし複数のルールが 1 つのグリッドサービスインスタンスに適用されるならば (すなわち、membershipContentRule SDE 値に含まれる複数の memberInterface を実装するならば)、全てのルールが満たされ**ねばならない (MUST)**。
  membershipContentRule が満たされることを保証するために、実装はどのインターフェイスがメンバーグリッドサービスインスタンスによって実装されるのか(内省や他の手段によって)知る必要があるだろうことを注意せよ。

```
<sd:serviceData
        name="membershipContentRule"
        type="MembershipContentRuleType"
        minOccurs="1"
        maxOccurs="unbounded"
        mutability="constant"
        modifiable="false"
        nillable=″false″ >
</sd:serviceData>

<xsd:complexType name="MembershipContentRuleType">
   <xsd:sequence>
      <xsd:element
           name="memberInterface"
```

```
            type="xsd:QName"
            minOccurs="1"
            maxOccurs="1"/>
      <xsd:element
            name="content"
            type="xsd:QName"
            minOccurs="0"
            maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
```

- entry

  この要素は、ServiceGroup の構成メンバーの構造体をサービスデータとして与える。ServiceGroup の各エントリに対して 1 つの entry SDE 値がある。個々の SDE 値は 3 つ組である。すなわち、このエントリを管理する ServiceGroupEntry サービスインスタンスを参照するロケータ、このエントリによって参照されるメンバーグリッドサービスインスタンスを参照するロケータ、この entry の content である。entry SDE の値と対応する ServiceGroupEntry の SDE の値は、membershipContentRule に従わ**なければならなく(MUST)**、(互いに) 一貫しなくて**よく(MAY)**、だが変更が無い状態に収束する**べきである (SHOULD)**。

```
<sd:serviceData
        name="entry"
        type="ogsi:EntryType"
        minOccurs="0"
        maxOccurs="unbounded"
        mutability="mutable"
        modifiable="false"
        nillable=”false” >
</sd:serviceData>
```

```
targetNamespace = “http://www.gridforum.org/namespaces/2003/03/OGSI”

<xsd:complexType name="EntryContentType">
   <xsd:sequence>
      <xsd:any namespace=”##any”
               minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="EntryType">
   <xsd:sequence>
      <xsd:element name=”serviceGroupEntryLocator”
              type="ogsi:LocatorType"
              minOccurs="1" maxOccurs="1"
              nillable="true"/>
      <xsd:element name=”memberServiceLocator”
```

```
            type="ogsi:LocatorType"
            minOccurs="1" maxOccurs="1"/>
      <xsd:element name="content"
            type="ogsi:EntryContentType"
            minOccurs="1" maxOccurs="1"/>
   </xsd:sequence>
</xsd:complexType>
```

13.1.2 ServiceGroup: 操作

ServiceGroup portType は操作を定義しない。GridService から継承される操作が、生存期間の管理や ServiceGroup の SDE の問い合わせに使われる**べきである(SHOULD)**。

13.2 *ServiceGroupEntry portType*

この portType は、ServiceGroup 内の個々のエントリを管理可能なインターフェイスを定義する。個々の ServiceGroupEntry サービスインスタンスは、その ServiceGroup のメンバーであるグリッドサービスインスタンスを参照する。ServiceGroupEntry サービスインスタンスの GSH を、そのエントリへのユニークなキーとして使用可能で**ある (MAY)**。このキーは、同じサービスインスタンスへの複数の参照が別個のエントリとして 1 つの ServiceGroup に含まれることを可能にする。これは、例えばそのサービスインスタンスの複数のプロパティを記録するためである。クライアントは、この情報を問い合わせるために findServiceData 操作を使用**可能である (MAY)**。

ServiceGroupEntry portType は GridService portType を拡張する。

13.2.1 ServiceGroupEntry: サービスデータ宣言

- memberServiceLocator

  このエントリが属するメンバーグリッドサービスインスタンスへのサービスロケータを持つ。このロケータは、このグリッドサービスインスタンスの生存期間中同じグリッドサービスインスタンスを参照し**なければならない (MUST)**。ただし、このロケータ内のハンドルと参照はそのサービスインスタンスの生存期間中に変化しても**よい (MAY)**。

```
<sd:serviceData
        name="memberServiceLocator"
        type="ogsi:LocatorType"
        minOccurs="1"
        maxOccurs="1"
        mutability="mutable"
        modifiable="false"
        nillable="false" >
</sd:serviceData>
```

- content

  これはメンバーサービスインスタンスに関するいくつかの情報を広告する XML 要素である。ここで、この ServiceGroupEntry を含む ServiceGroup portType の

membershipContentRule SDE を考える。Content 要素はこの SDE 中に（QName によって）列挙される XSD 要素宣言に従う。

```
<sd:serviceData
        name="content"
        type="ogsi:EntryContentType"
        minOccurs="1"
        maxOccurs="1"
        mutability="mutable"
        modifiable="false"
        nillable="false" >
</sd:serviceData>
```

13.2.2 ServiceGroupEntry: 操作

ServiceGroupEntry portType は操作を定義しない。GridService portType から継承される生存期間管理操作が、エントリサービスインスタンスの生存期間の管理に使われる**べきである（SHOULD)**、ServiceGroupEntry サービスインスタンスを破棄する際、ServiceGroup からメンバーサービスインスタンスを削除する**べきである（SHOULD)**。GridService portType から継承されるサービスデータ操作が、content SDE などの entry サービスインスタンスの SDE の問い合わせに用いられる**べきである（SHOULD)**。

13.3 *ServiceGroupRegistration portType*

ServiceGroupRegistration portType は、ServiceGroup のための管理インターフェイス(add や remove 操作)を提供する。

この portType は ServiceGroup portType を拡張する。

13.3.1 ServiceGroupRegistration: サービスデータ宣言

- addExtensibility

  add 操作の操作拡張性宣言(§7.8)の集合。クライアントは、この SDE の値によって宣言された任意の正規 inputElement を、add 操作の Content 引数として使用**可能である（MAY)**。また同 inputElement は、追加することの意味と返り値についての情報を提供する。これらの addExtensibility 要素は、ServiceGroup の membershipContentRule SDE に列挙される要素と異なっても**よい（MAY)**。add 操作は、Content 引数の値を membershipContentRule に含まれる要素に変換する。

```
<sd:serviceData
        name="addExtensibility"
        type="ogsi:OperationExtensibilityType"
        minOccurs="0"
        maxOccurs="unbounded"
        mutability="static"
        modifiable="false"
        nillable="false" >
</sd:serviceData>
```

OGSI は 1 つの適切な addExtensibility SDE 値、ogsi:EntryContentType を定義する。それは、entry 要素や ServiceGroup の ServiceGroupsEntries と同じ方法ですでに構造化された content の追加に使われることが**可能である(MAY)**。

- removeExtensibility

  remove 操作の操作拡張性宣言(§7.8)の集合。クライアントは、この SDE の値によって宣言された任意の正規 inputElement を、remove 操作の MatchExpression 引数として使用**可能である (MAY)**。また同 inputElement は、削除することの意味とこの remove 操作よる返り値についての情報を提供する。すべての一致式 (match expression) は、"ogsi:EntryType" 型の値を引数としてとる Boolean 値を返す関数で**なければならない(MUST)**。

```
<sd:serviceData
        name="removeExtensibility"
        type="ogsi:OperationExtensibilityType"
        minOccurs="1"
        maxOccurs="unbounded"
        mutability="static"
        modifiable="false"
        nillable="false" >
</sd:serviceData>
```

この portType は、全ての ServiceGroupRegistration サービスがサポートし**なければならない(MUST)**、1 つの標準 removeExtensibility SDE 値を定義し**なければならない(MUST)**。この matchByLocatorEquivalence 一致式は、字面の等価性比較を目的として、1 つ以上のロケータを、サービスグループに含まれる各 entry の memberServiceLocator の正規 XML 形式と比較する。この引数の XSD 定義は以下の通りである。

```
targetNamespace = "http://www.gridforum.org/namespaces/2003/03/OGSI"

<xsd:element name="matchByLocatorEquivalence"
             type="ogsi:MatchByLocatorEquivalenceType"/>

<xsd:complexType name="MatchByLocatorEquivalenceType">
   <xsd:sequence>
      <xsd:element name="locator" type="ogsi:LocatorType"
                   minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
```

この portType は以下の静的 SDE 値を定義し**なければならない(MUST)**。

```
<sd:staticServiceDataValues>
   <ogsi:removeExtensibility
         inputElement="ogsi:matchByLocatorEquivalence" />
</sd:staticServiceDataValues>
```

13.3.2 ServiceGroupRegistration:操作

13.3.2.1 ServiceGroupRegistration :: add

add 操作は ServiceGroupEntry を作成し、それを ServiceGroup に追加する。add 操作が失敗した場合、新規エントリは作成されない。失敗した add 操作の他のセマンティクスは、ServiceGroupRegistration を拡張する portType によって定義されても**よい (MAY)**。

入力:

- *ServiceLocator:* ServiceGroup に含まれることになるメンバーグリッドサービスインスタンスへの serviceLocator。これは、membershipContentRule の要素の 1 つに従う型を持つメンバーサービスインスタンスを参照し**なければならない (MUST)**。
- *Content:* サービスグループにおいて ServiceLocator と関連付ける content。この拡張可能な引数は、addExtensibility SDE 値の 1 つで表される inputElement 宣言に従わ**なければならない(MUST)**。このサービスインスタンスは、この引数のルート要素のタグに基づいて行うべき動作を推定する。ここで、serviceLocator 引数で与えられる型のグリッドサービスインスタンスに対する membershipContentRule SDE を考える。 Content は、サービスグループに固有のセマンティックスに従って、この SDE の 1 つ以上のエントリに従う要素へと処理される。結果として変換された Content は ServiceGroupEntry の Content SDE になる。
- *TerminationTime* (任意): 作成された ServiceGroupEntry サービスインスタンスの、クライアントが許容可能な最も早い初期終了時刻と最も遅い初期終了時刻。これらの値は、§7.6 で定義される terminationTime 要素として表現される。

出力:

- *ServiceLocator:* 新しく作成される ServiceGroupEntry への serviceLocator。
- *CurrentTerminationTime:* その ServiceGroupEntry サービスの現在の予定終了時刻を与える TerminationTimeType 型（§7.6 で定義）の要素。この CurrentTerminationTime の timestamp 属性は、その ServiceGroupEntry サービスが作成された時刻で**なければならない (MUST)**。

故障:

- *ExtensibilityNotSupportedFault:* Content の型がこのサービスインスタンスではサポートされていないため、インスタンスがその Content を評価できない。
- *ExtensibilityTypeFault*: Content として渡された値が、その型に違反する。
- *ContentCreationFailedFault*: 操作が、与えられた Content 引数と serviceLocator 引数から、有効な Content 要素（membershipContentRule SDE によって定義される）を作成できなかった。
- *UnsupportedMemberInterfaceFault*: ServiceLocator 引数によって参照されるメンバーサービスインスタンスの型が、membershipContentRule に従っていない。

- *AddRefusedFault*: ServiceGroupRegistration が、ServiceGroupRegistration (もしくはそのサブタイプ) のセマンティクスに基づいてメンバーサービスインスタンスの新しいエントリを作成することを拒否した。
- *Fault*: 全ての他の故障。

13.3.2.2 ServiceGroupRegistration :: remove

remove 操作は、入力式に一致する ServiceGroupEntry を ServiceGroup から削除する。

入力:

- *MatchExpression:* この拡張可能な引数は、removeExtensibility SDE 値の 1 つで表される inputElement 宣言に従わ**なければならない(MUST)**。このサービスインスタンスは、この引数のルート要素のタグに基づいて行うべき動作を推定する。MatchExpression は ServiceGroup 内の全てのエントリに対して評価される。一致する各 entry は ServiceGroup から削除される。削除される entry が ServiceGroupEntry サービスインスタンスを保持するなら、GridService::destroy 操作がそのインスタンスに送られる。そのような destroy 操作が完了する前に、この remove 操作は完了しても**よい (MAY)**。この操作は、entry が参照するメンバーグリッドサービスには影響をもたらさない。

出力:

- 操作完了の通知以外に、無し。

故障:

- *ExtensibilityNotSupportedFault:* MatchExpression の型がこのサービスインスタンスではサポートされていないため、インスタンスがその MatchExpression を評価できない。
- *ExtensibilityTypeFault*: MatchExpression として渡された値が、その型に違反する。
- *MatchFailedFault*: MatchExpression に一致するエントリが無かった。
- *RemoveFailedFault*: 一致は見られたが、削除が他の理由のため失敗した。
- *Fault*: 他の全ての故障。

14 セキュリティに関する考察

本仕様は、グリッドサービスインスタンスとクライアントとの抽象相互作用 (abstract interaction) を定義するものである。そのような相互作用は安全でなければならないと仮定されているが、セキュリティの詳細は本仕様の範囲外である。セキュリティは、抽象相互作用の特定の通信プロトコルへの束縛方法、サービス挙動のポリシー管理インターフェイスを通じた特化方法、セキュリティ機能を特定のプログラミング環境で利用可能にする方法を定義する関連した仕様書で取り組まれるべきである。

15 編者情報

Steven Tuecke
Distributed Systems Laboratory
Mathematics and Computer Science Division
Argonne National Laboratory
Argonne, IL 60439
Phone: 630-252-8711
Email: tuecke@mcs.anl.gov

Karl Czajkowski
Information Sciences Institute
University of Southern California
Marina del Rey, CA 90292
Email: karlcz@isi.edu

Ian Foster
Argonne National Laboratory & University of Chicago
Argonne, IL 60439
Email: foster@mcs.anl.gov

Jeffrey Frey
IBM
Poughkeepsie, NY 12601
Email: jafrey@us.ibm.com

Steve Graham
IBM
4400 Silicon Drive
Research Triangle Park, NC, 27713
Email: sggraham@us.ibm.com

Carl Kesselman
Information Sciences Institute
University of Southern California
Marina del Rey, CA 90292
Email: carl@isi.edu

Tom Maquire
IBM
Poughkeepsie, NY 12601
Email: tmaguire@us.ibm.com

Thomas Sandholm
Argonne National Laboratory
Argonne, IL 60439
Email: sandholm@mcs.anl.gov

Dr. David Snelling
Fujitsu Laboratories of Europe

Hayes Park Central
Hayes End Road
Hayes, Middlesex UB4 8FE
UK
Email: d.snelling@fle.fujitsu.com

Peter Vanderbilt
NASA Ames Research Center
Moffett Field, CA 94035-1000
Email: pv@nas.nasa.gov

16 貢献者

次に挙げる人々による、本仕様への貢献を大変感謝する。

Nick Butler, Donald Ferguson, Andrew Grimshaw, Shel Finkelstein, Frank Leymann, Martin Nally, Jeff Nick, John Rofrano, Ellen Stokes, Tony Storey, Jay Unger, Sanjiva Weerawarana

17 謝辞

本文書について議論して頂いた多くの同僚に大変感謝する。特に、Malcolm Atkinson、Tim Banks、Ed Boden,、Brian Carpenter、Francisco Curbera、Dennis Gannon、Marty Humphrey、Keith Jackson、Bill Johnston、Kate Keahey、Lee Liming、Miron Livny、Sastry Malladi,、Savas Parastatidis、Norman Paton、Jean-Pierre Prost、Frank Siebenlist、Scott Sylvester、Gregor von Laszewski、Von Welch、そして Mike Williams に感謝する(アルファベット順。名前を挙げ落としてしまった人におわびする)。

この仕事の一部は、IBM、the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, U.S. Department of Energy (Contract W-31-109-Eng-38 と DE-AC03-76SF0098)、the National Science Foundation、the NASA Information Power Grid project による補助を受けた。

18 参考文献

18.1 仕様書等

[RFC 2119]

Key words for use in RFCs to Indicate Requirement Levels, S. Bradner, Author. Internet Engineering Task Force, RFC 2119, March 1997. Available at http://www.ietf.org/rfc/rfc2119.txt

[WSDL 1.2]

Web Services Description Language (WSDL) Version 1.2, Published W3C Working Draft, World Wide Web Consortium. Available at http://www.w3.org/TR/wsdl12/

[WSDL 1.2 DRAFT]

Web Services Description Language (WSDL) Version 1.2, W3C Working Draft 3 March 2003, World Wide Web Consortium. Available at http://www.w3.org/TR/2003/WD-wsdl12-20030303

18.2 参考情報

[Globus Overview]

Globus: A Toolkit-Based Grid Architecture, I. Foster, C. Kesselman. In [Grid Book], 259-278.

[Grid Anatomy]

The Anatomy of the Grid: Enabling Scalable Virtual Organizations, I. Foster, C. Kesselman, S. Tuecke. International Journal of High Performance Computing Applications, 15 (3). 200-222. 2001. Available at http://www.globus.org/research/papers/anatomy.pdf

[Grid Book]

The Grid: Blueprint for a New Computing Infrastructure, I. Foster, C. Kesselman, eds. Morgan Kaufmann, 1999.

[Grid Physiology]

The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration, I. Foster, C. Kesselman, J. Nick, S. Tuecke. Globus Project, 2002. Available at http://www.globus.org/research/papers/ogsa.pdf

[JAX-RPC]

Java™ API for XML-Based RPC (JAX-RPC). *http://java.sun.com/xml/jaxrpc/docs.html*

[Web Services Book]

Building Web Services with Java: Making Sense of XML, SOAP, WSDL, and UDDI, s. Graham, S. Simeonov, T. Boubez, G. Daniels, D. Davis, Y. Nakamura, R. Neyama. Sams, 2001.

[WSIF]

Welcome to WSIF: Web Services Invocation Framework, http://www.apache.org/wsif/

19 正規 XSD と WSDL 仕様

本節では、本文書で説明したすべてについて、完全な正規の XSD と WSDL 定義を挙げる。本節で挙げる定義と他の節での説明に不一致がある場合、本節の定義が正規の定義であるとみなさ**なければならない (MUST)**。

19.1 *http://www.gridforum.org/namespaces/2003/03/OGSI*

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="OGSI"
             targetNamespace="http://www.gridforum.org/namespaces/2003/03/OGSI"
             xmlns:ogsi="http://www.gridforum.org/namespaces/2003/03/OGSI"

xmlns:gwsdl="http://www.gridforum.org/namespaces/2003/03/gridWSDLExtensions"
             xmlns:sd="http://www.gridforum.org/namespaces/2003/03/serviceData"
             xmlns:xsd="http://www.w3.org/2001/XMLSchema"
             xmlns="http://schemas.xmlsoap.org/wsdl/">

<types>
  <schema targetNamespace="http://www.gridforum.org/namespaces/2003/03/OGSI"
          xmlns="http://www.w3.org/2001/XMLSchema"
          attributeFormDefault="qualified"
          elementFormDefault="qualified">
    <!-- Common Types -->
    <simpleType name="ExtendedDateTimeType">
      <union memberTypes="ogsi:InfinityType xsd:dateTime"/>
    </simpleType>

    <simpleType name="InfinityType">
      <restriction base="string">
        <enumeration value="infinity"/>
      </restriction>
    </simpleType>

    <attribute name="goodFrom" type="ogsi:ExtendedDateTimeType"/>
    <attribute name="goodUntil" type="ogsi:ExtendedDateTimeType"/>
    <attribute name="availableUntil"
        type="ogsi:ExtendedDateTimeType"/>

    <attributeGroup name="LifeTimePropertiesGroup">
      <attribute ref="ogsi:goodFrom" use="optional"/>
      <attribute ref="ogsi:goodUntil" use="optional"/>
      <attribute ref="ogsi:availableUntil" use="optional"/>
    </attributeGroup>

    <element name="reference" type="ogsi:ReferenceType"/>
    <complexType name="ReferenceType" abstract="true">
      <attribute ref="ogsi:goodFrom" use="optional"/>
      <attribute ref="ogsi:goodUntil" use="optional"/>
    </complexType>

    <!-- The content of this type MUST be a wsdl:definitions element
    with a single wsdl:service child element -->
    <complexType name="WSDLReferenceType">
      <complexContent>
        <extension base="ogsi:ReferenceType">
```

```
        <sequence>
          <any namespace="http://schemas.xmlsoap.org/wsdl/"
                minOccurs="1" maxOccurs="1" processContents="lax"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>

  <element name="handle" type="ogsi:HandleType"/>
  <simpleType name="HandleType">
    <restriction base="anyURI"/>
  </simpleType>

  <element name="locator" type="ogsi:LocatorType"/>
  <complexType name="LocatorType">
    <sequence>
        <element ref="ogsi:handle" minOccurs="0"
            maxOccurs="unbounded"/>
        <element ref="ogsi:reference" minOccurs="0"
            maxOccurs="unbounded"/>
        <element name="interface" type="QName" minOccurs="0"
            maxOccurs="unbounded"/>
    </sequence>
  </complexType>


  <!-- Grid Service Types -->
  <complexType name="ExtensibilityType">
    <sequence>
      <any namespace="##any"/>
    </sequence>
  </complexType>

  <!-- Grid Service Service Data Types -->
  <complexType name="OperationExtensibilityType">
    <attribute name="inputElement" type="QName" use="optional"/>
  </complexType>

  <complexType name="TerminationTimeType">
    <attribute name="after" type="ogsi:ExtendedDateTimeType"
          use="optional"/>
    <attribute name="before" type="ogsi:ExtendedDateTimeType"
          use="optional"/>
    <attribute name="timestamp" type="dateTime" use="optional"/>
  </complexType>

  <element name="queryByServiceDataNames" type="ogsi:QNamesType"/>

  <element name="deleteByServiceDataNames" type="ogsi:QNamesType"/>
```

```
<element name="setByServiceDataNames"
    type="ogsi:ExtensibilityType"/>

<complexType name="QNamesType">
  <sequence>
    <element name="name" type="QName" minOccurs="0"
          maxOccurs="unbounded"/>
  </sequence>
</complexType>

<!-- Grid Service Message Types -->
<element name="findServiceData">
  <complexType>
    <sequence>
      <element name="queryExpression"
            type="ogsi:ExtensibilityType"/>
    </sequence>
  </complexType>
</element>
<element name="findServiceDataResponse">
  <complexType>
    <sequence>
      <element name="result" type="ogsi:ExtensibilityType"/>
    </sequence>
  </complexType>
</element>
<element name="setServiceData">
  <complexType>
    <sequence>
      <element name="updateExpression"
            type="ogsi:ExtensibilityType"/>
    </sequence>
  </complexType>
</element>
<element name="setServiceDataResponse">
  <complexType>
    <sequence>
      <element name="result" type="ogsi:ExtensibilityType"/>
    </sequence>
  </complexType>
</element>
<element name="requestTerminationBefore">
  <complexType>
    <sequence>
      <element name="terminationTime"
            type="ogsi:ExtendedDateTimeType"/>
    </sequence>
  </complexType>
```

```
    </element>
    <element name="requestTerminationBeforeResponse">
      <complexType>
        <sequence>
          <element name="currentTerminationTime"
                type="ogsi:TerminationTimeType"/>
        </sequence>
      </complexType>
    </element>
    <element name="requestTerminationAfter">
      <complexType>
        <sequence>
          <element name="terminationTime"
                type="ogsi:ExtendedDateTimeType"/>
        </sequence>
      </complexType>
    </element>
    <element name="requestTerminationAfterResponse">
      <complexType>
        <sequence>
          <element name="currentTerminationTime"
                type="ogsi:TerminationTimeType"/>
        </sequence>
      </complexType>
    </element>
    <element name="destroy">
      <complexType/>
    </element>
    <element name="destroyResponse">
      <complexType/>
    </element>

    <!-- Grid Service Fault Types -->
    <element name="fault" type="ogsi:FaultType"/>
    <complexType name="FaultType">
      <sequence>
        <element name="description"
                 type="string"
                 minOccurs="0"
                 maxOccurs="unbounded"/>
        <element name="originator"
                 type="ogsi:LocatorType"
                 minOccurs="1"
                 maxOccurs="1"/>
        <element name="timestamp"
                 type="dateTime"
                 minOccurs="1"
                 maxOccurs="1"/>
        <element name="faultcause"
```

```
                type="ogsi:FaultType"
                minOccurs="0"
                maxOccurs="unbounded"/>
      <element name="faultcode"
                type="ogsi:FaultCodeType"
                minOccurs="0"
                maxOccurs="1"/>
      <element name="extension"
                type="ogsi:ExtensibilityType"
                minOccurs="0"
                maxOccurs="1"/>
    </sequence>
  </complexType>
  <complexType name="FaultCodeType">
    <simpleContent>
      <extension base="string">
        <attribute name="faultscheme" type="anyURI" use="required"/>
      </extension>
    </simpleContent>
  </complexType>
  <element name="serviceNotDestroyedFault"
      type="ogsi:ServiceNotDestroyedFaultType"/>
  <complexType name="ServiceNotDestroyedFaultType">
    <complexContent>
      <extension base="ogsi:FaultType"/>
    </complexContent>
  </complexType>
  <element name="extensibilityTypeFault"
      type="ogsi:ExtensibilityTypeFaultType"/>
  <complexType name="ExtensibilityTypeFaultType">
    <complexContent>
      <extension base="ogsi:FaultType"/>
    </complexContent>
  </complexType>
  <element name="extensibilityNotSupportedFault"
      type="ogsi:ExtensibilityNotSupportedFaultType"/>
  <complexType name="ExtensibilityNotSupportedFaultType">
    <complexContent>
      <extension base="ogsi:FaultType"/>
    </complexContent>
  </complexType>
  <element name="targetInvalidFault"
      type="ogsi:TargetInvalidFaultType"/>
  <complexType name="TargetInvalidFaultType">
    <complexContent>
      <extension base="ogsi:FaultType"/>
    </complexContent>
  </complexType>
  <element name="cardinalityViolationFault"
```

```
      type="ogsi:CardinalityViolationFaultType"/>
  <complexType name="CardinalityViolationFaultType">
    <complexContent>
      <extension base="ogsi:FaultType"/>
    </complexContent>
  </complexType>
  <element name="mutabilityViolationFault"
      type="ogsi:MutabilityViolationFaultType"/>
  <complexType name="MutabilityViolationFaultType">
    <complexContent>
      <extension base="ogsi:FaultType"/>
    </complexContent>
  </complexType>
  <element name="modifiabilityViolationFault"
      type="ogsi:ModifiabilityViolationFaultType"/>
  <complexType name="ModifiabilityViolationFaultType">
    <complexContent>
      <extension base="ogsi:FaultType"/>
    </complexContent>
  </complexType>
  <element name="typeViolationFault"
      type="ogsi:TypeViolationFaultType"/>
  <complexType name="TypeViolationFaultType">
    <complexContent>
      <extension base="ogsi:FaultType"/>
    </complexContent>
  </complexType>
  <element name="incorrectValueFault"
      type="ogsi:IncorrectValueFaultType"/>
  <complexType name="IncorrectValueFaultType">
    <complexContent>
      <extension base="ogsi:FaultType"/>
    </complexContent>
  </complexType>
  <element name="partialFailureFault"
      type="ogsi:PartialFailureFaultType"/>
  <complexType name="PartialFailureFaultType">
    <complexContent>
      <extension base="ogsi:FaultType">
        <sequence>
          <element name="failedServiceData" type="ogsi:QNamesType"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <element name="terminationTimeUnchangedFault"
      type="ogsi:TerminationTimeUnchangedFaultType"/>
  <complexType name="TerminationTimeUnchangedFaultType">
    <complexContent>
```

```
      <extension base="ogsi:FaultType"/>
    </complexContent>
  </complexType>

  <!-- Handle Resolver Message Types -->
  <element name="findByHandle">
    <complexType>
      <sequence>
        <element name="handleSet" type="ogsi:LocatorType"/>
        <element name="gsrExclusionSet" type="ogsi:LocatorType"
            minOccurs="0" maxOccurs="1"/>
      </sequence>
    </complexType>
  </element>
  <element name="findByHandleResponse">
    <complexType>
      <sequence>
        <element ref="ogsi:locator"/>
      </sequence>
    </complexType>
  </element>

  <!-- Handle Resolver Fault Types -->
  <element name="invalidHandleFault"
      type="ogsi:InvalidHandleFaultType"/>
  <complexType name="InvalidHandleFaultType">
    <complexContent>
      <extension base="ogsi:FaultType"/>
    </complexContent>
  </complexType>
  <element name="noAdditionalReferencesAvailableFault"
      type="ogsi:NoAdditionalReferencesAvailableFaultType"/>
  <complexType name="NoAdditionalReferencesAvailableFaultType">
    <complexContent>
      <extension base="ogsi:FaultType"/>
    </complexContent>
  </complexType>
  <element name="noReferencesAvailableFault"
      type="ogsi:NoReferencesAvailableFaultType"/>
  <complexType name="NoReferencesAvailableFaultType">
    <complexContent>
      <extension base="ogsi:FaultType"/>
    </complexContent>
  </complexType>
  <element name="redirectionFault"
      type="ogsi:RedirectionFaultType"/>
  <complexType name="RedirectionFaultType">
    <complexContent>
      <extension base="ogsi:FaultType">
```

```
        <sequence>
          <element ref="ogsi:locator"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <element name="noSuchServiceFault"
      type="ogsi:NoSuchServiceFaultType"/>
  <complexType name="NoSuchServiceFaultType">
    <complexContent>
      <extension base="ogsi:NoReferencesAvailableFaultType"/>
    </complexContent>
  </complexType>
  <element name="noSuchServiceStartedFault"
      type="ogsi:NoSuchServiceStartedFaultType"/>
  <complexType name="NoSuchServiceStartedFaultType">
    <complexContent>
      <extension base="ogsi:NoReferencesAvailableFaultType"/>
    </complexContent>
  </complexType>
  <element name="serviceHasTerminatedFault"
      type="ogsi:ServiceHasTerminatedFaultType"/>
  <complexType name="ServiceHasTerminatedFaultType">
    <complexContent>
      <extension base="ogsi:NoReferencesAvailableFaultType"/>
    </complexContent>
  </complexType>
  <element name="temporarilyUnavailableFault"
      type="ogsi:TemporarilyUnavailableFaultType"/>
  <complexType name="TemporarilyUnavailableFaultType">
    <complexContent>
      <extension base="ogsi:NoReferencesAvailableFaultType">
        <sequence>
          <element name="available" type="dateTime"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>

  <!-- Notification Source Service Data Types -->
  <element name="subscribeByServiceDataNames"
      type="ogsi:SubscribeByNameType"/>
  <complexType name="SubscribeByNameType">
    <complexContent>
      <extension base="ogsi:QNamesType">
        <attribute name="minInterval" type="duration"
            use="optional"/>
        <attribute name="maxInterval" type="ogsi:MaxIntervalType"/>
      </extension>
```

```
    </complexContent>
  </complexType>

  <simpleType name="MaxIntervalType">
    <union memberTypes="ogsi:InfinityType xsd:duration"/>
  </simpleType>

  <!-- Notification Source Message Types -->
  <element name="subscribe">
    <complexType>
      <sequence>
        <element name="subscriptionExpression"
            type="ogsi:ExtensibilityType"/>
        <element name="sink" type="ogsi:LocatorType"/>
        <element name="expirationTime"
            type="ogsi:ExtendedDateTimeType"/>
      </sequence>
    </complexType>
  </element>
  <element name="subscribeResponse">
    <complexType>
      <sequence>
        <element name="subscriptionInstanceLocator"
            type="ogsi:LocatorType"/>
        <element name="currentTerminationTime"
            type="ogsi:TerminationTimeType"/>
      </sequence>
    </complexType>
  </element>

  <!-- Notification Sink Message Types -->
  <element name="deliverNotification">
    <complexType>
      <sequence>
        <element name="message" type="ogsi:ExtensibilityType"/>
      </sequence>
    </complexType>
  </element>

  <!-- Factory Service Data Types -->
  <complexType name="CreateServiceExtensibilityType">
    <complexContent>
      <extension base="ogsi:OperationExtensibilityType">
        <sequence>
          <element name="createsInterface" type="QName"
              minOccurs="1" maxOccurs="unbounded"/>
        </sequence>
      </extension>
    </complexContent>
```

```
    </complexType>

    <!-- Factory Message Types -->
    <element name="createService">
      <complexType>
        <sequence>
          <element name="terminationTime"
              type="ogsi:TerminationTimeType" minOccurs="0"
              maxOccurs="1"/>
          <element name="creationParameters"
              type="ogsi:ExtensibilityType" minOccurs="0"
              maxOccurs="1"/>
        </sequence>
      </complexType>
    </element>
    <element name="createServiceResponse">
      <complexType>
        <sequence>
          <element name="locator" type="ogsi:LocatorType"/>
          <element name="currentTerminationTime"
              type="ogsi:TerminationTimeType"/>
          <element name="extensibilityOutput"
              type="ogsi:ExtensibilityType" minOccurs="0"
              maxOccurs="1"/>
        </sequence>
      </complexType>
    </element>

    <!-- Factory Fault Types -->
    <element name="serviceAlreadyExistsFault"
        type="ogsi:ServiceAlreadyExistsFaultType"/>
    <complexType name="ServiceAlreadyExistsFaultType">
      <complexContent>
        <extension base="ogsi:FaultType">
          <sequence>
            <element name="existingService" type="ogsi:LocatorType"/>
          </sequence>
        </extension>
      </complexContent>
    </complexType>

    <!-- Service Group Service Data Types -->
    <complexType name="MembershipContentRuleType">
      <sequence>
        <element name="memberInterface"
                 type="QName"
                 minOccurs="1"
                 maxOccurs="1"/>
        <element name="content"
```

```
              type="QName"
              minOccurs="0"
              maxOccurs="unbounded"/>
    </sequence>
  </complexType>
  <complexType name="EntryContentType">
    <sequence>
      <any namespace="##any" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
  </complexType>

  <complexType name="EntryType">
    <sequence>
      <element name="serviceGroupEntryLocator"
              type="ogsi:LocatorType"
              minOccurs="1"
              maxOccurs="1"
              nillable="true"/>
      <element name="memberServiceLocator"
              type="ogsi:LocatorType"
              minOccurs="1"
              maxOccurs="1"/>
      <element name="content"
          type="ogsi:EntryContentType"
              minOccurs="1"
              maxOccurs="1"/>
    </sequence>
  </complexType>


  <!-- Service Group Registration Service Data Types -->
  <element name="matchByLocatorEquivalence"
      type="ogsi:MatchByLocatorEquivalenceType"/>
  <complexType name="MatchByLocatorEquivalenceType">
    <sequence>
      <element name="locator" type="ogsi:LocatorType" minOccurs="0"
          maxOccurs="unbounded"/>
    </sequence>
  </complexType>

  <!-- Service Group Registration Message Types -->
  <element name="add">
    <complexType>
      <sequence>
        <element name="serviceLocator" type="ogsi:LocatorType"/>
        <element name="content" type="ogsi:ExtensibilityType"/>
        <element name="terminationTime"
            type="ogsi:TerminationTimeType" minOccurs="0"
            maxOccurs="1"/>
```

```
      </sequence>
    </complexType>
  </element>
  <element name="addResponse">
    <complexType>
      <sequence>
        <element name="serviceLocator" type="ogsi:LocatorType"/>
        <element name="currentTerminationTime"
            type="ogsi:TerminationTimeType"/>
      </sequence>
    </complexType>
  </element>
  <element name="remove">
    <complexType>
      <sequence>
        <element name="matchExpression"
            type="ogsi:ExtensibilityType"/>
      </sequence>
    </complexType>
  </element>
  <element name="removeResponse">
    <complexType/>
  </element>

  <!-- Service Group Registration Fault Types -->
  <element name="contentCreationFailedFault"
      type="ogsi:ContentCreationFailedFaultType"/>
  <complexType name="ContentCreationFailedFaultType">
    <complexContent>
      <extension base="ogsi:FaultType"/>
    </complexContent>
  </complexType>
  <element name="unsupportedMemberInterfaceFault"
      type="ogsi:UnsupportedMemberInterfaceFaultType"/>
  <complexType name="UnsupportedMemberInterfaceFaultType">
    <complexContent>
      <extension base="ogsi:FaultType"/>
    </complexContent>
  </complexType>
  <element name="addRefusedFault" type="ogsi:AddRefusedFaultType"/>
  <complexType name="AddRefusedFaultType">
    <complexContent>
      <extension base="ogsi:FaultType"/>
    </complexContent>
  </complexType>
  <element name="matchFailedFault"
      type="ogsi:MatchFailedFaultType"/>
  <complexType name="MatchFailedFaultType">
    <complexContent>
```

```
        <extension base="ogsi:FaultType"/>
      </complexContent>
    </complexType>
    <element name="removeFailedFault"
        type="ogsi:RemoveFailedFaultType"/>
    <complexType name="RemoveFailedFaultType">
      <complexContent>
        <extension base="ogsi:FaultType"/>
      </complexContent>
    </complexType>
  </schema>
</types>

<!-- Grid Service Messages -->
<message name="FindServiceDataInputMessage">
  <part name="parameters" element="ogsi:findServiceData"/>
</message>
<message name="FindServiceDataOutputMessage">
  <part name="parameters" element="ogsi:findServiceDataResponse"/>
</message>
<message name="SetServiceDataInputMessage">
  <part name="parameters" element="ogsi:setServiceData"/>
</message>
<message name="SetServiceDataOutputMessage">
  <part name="parameters" element="ogsi:setServiceDataResponse"/>
</message>
<message name="RequestTerminationBeforeInputMessage">
  <part name="parameters" element="ogsi:requestTerminationBefore"/>
</message>
<message name="RequestTerminationBeforeOutputMessage">
  <part name="parameters"
      element="ogsi:requestTerminationBeforeResponse"/>
</message>
<message name="RequestTerminationAfterInputMessage">
  <part name="parameters" element="ogsi:requestTerminationAfter"/>
</message>
<message name="RequestTerminationAfterOutputMessage">
  <part name="parameters"
      element="ogsi:requestTerminationAfterResponse"/>
</message>
<message name="DestroyInputMessage">
  <part name="parameters" element="ogsi:destroy"/>
</message>
<message name="DestroyOutputMessage">
  <part name="parameters" element="ogsi:destroyResponse"/>
</message>

<!-- Grid Service Fault Messages -->
<message name="FaultMessage">
```

```
  <part name="fault" element="ogsi:fault"/>
</message>
<message name="ServiceNotDestroyedFaultMessage">
  <part name="fault" element="ogsi:serviceNotDestroyedFault"/>
</message>
<message name="ExtensibilityTypeFaultMessage">
  <part name="fault" element="ogsi:extensibilityTypeFault"/>
</message>
<message name="ExtensibilityNotSupportedFaultMessage">
  <part name="fault" element="ogsi:extensibilityNotSupportedFault"/>
</message>
<message name="TargetInvalidFaultMessage">
  <part name="fault" element="ogsi:targetInvalidFault"/>
</message>
<message name="CardinalityViolationFaultMessage">
  <part name="fault" element="ogsi:cardinalityViolationFault"/>
</message>
<message name="MutabilityViolationFaultMessage">
  <part name="fault" element="ogsi:mutabilityViolationFault"/>
</message>
<message name="ModifiabilityViolationFaultMessage">
  <part name="fault" element="ogsi:modifiabilityViolationFault"/>
</message>
<message name="TypeViolationFaultMessage">
  <part name="fault" element="ogsi:typeViolationFault"/>
</message>
<message name="IncorrectValueFaultMessage">
  <part name="fault" element="ogsi:incorrectValueFault"/>
</message>
<message name="PartialFailureFaultMessage">
  <part name="fault" element="ogsi:partialFailureFault"/>
</message>
<message name="TerminationTimeUnchangedFaultMessage">
  <part name="fault" element="ogsi:terminationTimeUnchangedFault"/>
</message>

<!-- HandleResolver Messages -->
<message name="FindByHandleInputMessage">
  <part name="parameters" element="ogsi:findByHandle"/>
</message>
<message name="FindByHandleOutputMessage">
  <part name="parameters" element="ogsi:findByHandleResponse"/>
</message>

<!-- HandleResolver Fault Messages -->
<message name="InvalidHandleFaultMessage">
  <part name="fault" element="ogsi:invalidHandleFault"/>
</message>
<message name="NoAdditionalReferencesAvailableFaultMessage">
```

```
  <part name="fault"
      element="ogsi:noAdditionalReferencesAvailableFault"/>
</message>
<message name="NoReferencesAvailableFaultMessage">
  <part name="fault" element="ogsi:noReferencesAvailableFault"/>
</message>
<message name="RedirectionFaultMessage">
  <part name="fault" element="ogsi:redirectionFault"/>
</message>
<message name="NoSuchServiceFaultMessage">
  <part name="fault" element="ogsi:noSuchServiceFault"/>
</message>
<message name="NoSuchServiceStartedFaultMessage">
  <part name="fault" element="ogsi:noSuchServiceStartedFault"/>
</message>
<message name="ServiceHasTerminatedFaultMessage">
  <part name="fault" element="ogsi:serviceHasTerminatedFault"/>
</message>
<message name="TemporarilyUnavailableFaultMessage">
  <part name="fault" element="ogsi:temporarilyUnavailableFault"/>
</message>

<!-- Factory Messages -->
<message name="CreateServiceInputMessage">
  <part name="parameters" element="ogsi:createService"/>
</message>
<message name="CreateServiceOutputMessage">
  <part name="parameters" element="ogsi:createServiceResponse"/>
</message>

<!-- Factory Fault Messages -->
<message name="ServiceAlreadyExistsFaultMessage">
  <part name="fault" element="ogsi:serviceAlreadyExistsFault"/>
</message>

<!-- NotificaitonSource Messages -->
<message name="SubscribeInputMessage">
  <part name="parameters" element="ogsi:subscribe"/>
</message>
<message name="SubscribeOutputMessage">
  <part name="parameters" element="ogsi:subscribeResponse"/>
</message>

<!-- NotificationSink Messages -->
<message name="DeliverNotificationInputMessage">
  <part name="parameters" element="ogsi:deliverNotification"/>
</message>

<!-- ServiceGroupRegistration Messages -->
```

```
<message name="AddInputMessage">
  <part name="parameters" element="ogsi:add"/>
</message>
<message name="AddOutputMessage">
  <part name="parameters" element="ogsi:addResponse"/>
</message>
<message name="removeInputMessage">
  <part name="parameters" element="ogsi:remove"/>
</message>
<message name="removeOutputMessage">
  <part name="parameters" element="ogsi:removeResponse"/>
</message>

<!-- ServiceGroupRegistration Fault Messages -->
<message name="ContentCreationFailedFaultMessage">
  <part name="faults" element="ogsi:contentCreationFailedFault"/>
</message>
<message name="UnsupportedMemberInterfaceFaultMessage">
  <part name="faults" element="ogsi:unsupportedMemberInterfaceFault"/>
</message>
<message name="AddRefusedFaultMessage">
  <part name="faults" element="ogsi:addRefusedFault"/>
</message>
<message name="MatchFailedFaultMessage">
  <part name="faults" element="ogsi:matchFailedFault"/>
</message>
<message name="RemoveFailedFaultMessage">
  <part name="faults" element="ogsi:removeFailedFault"/>
</message>

<!-- Grid Service Port Type -->
<gwsdl:portType name="GridService">
  <operation name="findServiceData">
    <input message="ogsi:FindServiceDataInputMessage"/>
    <output message="ogsi:FindServiceDataOutputMessage"/>
    <fault name="ExtensibilityNotSupportedFault"
        message="ogsi:ExtensibilityNotSupportedFaultMessage"/>
    <fault name="ExtensibilityTypeFault"
        message="ogsi:ExtensibilityTypeFaultMessage"/>
    <fault name="TargetInvalidFault"
        message="ogsi:TargetInvalidFaultMessage"/>
    <fault name="Fault" message="ogsi:FaultMessage"/>
  </operation>
  <operation name="setServiceData">
    <input message="ogsi:SetServiceDataInputMessage"/>
    <output message="ogsi:SetServiceDataOutputMessage"/>
    <fault name="ExtensibilityNotSupportedFault"
        message="ogsi:ExtensibilityNotSupportedFaultMessage"/>
    <fault name="ExtensibilityTypeFault"
```

```
        message="ogsi:ExtensibilityTypeFaultMessage"/>
    <fault name="CadinalityViolationFault"
        message="ogsi:CardinalityViolationFaultMessage"/>
    <fault name="MutabilityViolationFault"
        message="ogsi:MutabilityViolationFaultMessage"/>
    <fault name="ModifabilityViolationFault"
        message="ogsi:ModifiabilityViolationFaultMessage"/>
    <fault name="TypeViolationFault"
        message="ogsi:TypeViolationFaultMessage"/>
    <fault name="IncorrectValueFault"
        message="ogsi:IncorrectValueFaultMessage"/>
    <fault name="PartialFailureFault"
        message="ogsi:PartialFailureFaultMessage"/>
    <fault name="Fault" message="ogsi:FaultMessage"/>
  </operation>
  <operation name="requestTerminationAfter">
    <input message="ogsi:RequestTerminationAfterInputMessage"/>
    <output message="ogsi:RequestTerminationAfterOutputMessage"/>
    <fault name="TerminationTimeUnchangedFault"
        message="ogsi:TerminationTimeUnchangedFaultMessage"/>
    <fault name="Fault" message="ogsi:FaultMessage"/>
  </operation>
  <operation name="requestTerminationBefore">
    <input message="ogsi:RequestTerminationBeforeInputMessage"/>
    <output message="ogsi:RequestTerminationBeforeOutputMessage"/>
    <fault name="TerminationTimeUnchangedFault"
        message="ogsi:TerminationTimeUnchangedFaultMessage"/>
    <fault name="Fault" message="ogsi:FaultMessage"/>
  </operation>
  <operation name="destroy">
    <input message="ogsi:DestroyInputMessage"/>
    <output message="ogsi:DestroyOutputMessage"/>
    <fault name="ServiceNotDestroyedFault"
        message="ogsi:ServiceNotDestroyedFaultMessage"/>
    <fault name="Fault" message="ogsi:FaultMessage"/>
  </operation>
  <sd:serviceData name="interface"
                  type="xsd:QName"
                  minOccurs="1"
                  maxOccurs="unbounded"
                  mutability="constant"
                  modifiable="false"
                  nillable="false"/>
  <sd:serviceData name="serviceDataName"
                  type="xsd:QName"
                  minOccurs="0"
                  maxOccurs="unbounded"
                  mutability="mutable"
                  modifiable="false"
```

```
                nillable="false"/>
 <sd:serviceData name="factoryLocator"
                type="ogsi:LocatorType"
                minOccurs="1"
                maxOccurs="1"
                mutability="mutable"
                modifiable="false"
                nillable="true"/>
 <sd:serviceData name="gridServiceHandle"
                type="ogsi:HandleType"
                minOccurs="0"
                maxOccurs="unbounded"
                mutability="extendable"
                modifiable="false"
                nillable="false"/>
 <sd:serviceData name="gridServiceReference"
                type="ogsi:ReferenceType"
                minOccurs="1"
                maxOccurs="unbounded"
                mutability="mutable"
                modifiable="false"
                nillable="false"/>
 <sd:serviceData name="findServiceDataExtensibility"
                type="ogsi:OperationExtensibilityType"
                minOccurs="1"
                maxOccurs="unbounded"
                mutability="static"
                modifiable="false"
                nillable="false"/>
 <sd:serviceData name="setServiceDataExtensibility"
                type="ogsi:OperationExtensibilityType"
                minOccurs="1"
                maxOccurs="unbounded"
                mutability="static"
                modifiable="false"
                nillable="false"/>
 <sd:serviceData name="terminationTime"
                type="ogsi:TerminationTimeType"
                minOccurs="1"
                maxOccurs="1"
                mutability="mutable"
                modifiable="false"
                nillable="false"/>
 <sd:staticServiceDataValues>
   <ogsi:findServiceDataExtensibility
       inputElement="ogsi:queryByServiceDataNames"/>
   <ogsi:setServiceDataExtensibility
       inputElement="ogsi:setByServiceDataNames"/>
   <ogsi:setServiceDataExtensibility
```

```
        inputElement="ogsi:deleteByServiceDataNames"/>
  </sd:staticServiceDataValues>
</gwsdl:portType>


<!-- HandleResolver Port Type -->
<gwsdl:portType name="HandleResolver" extends="ogsi:GridService">
  <operation name="findByHandle">
    <input message="ogsi:FindByHandleInputMessage"/>
    <output message="ogsi:FindByHandleOutputMessage"/>
    <fault name="InvalidHandleFault"
        message="ogsi:InvalidHandleFaultMessage"/>
    <fault name="NoAdditionalReferencesAvailableFault"
        message="ogsi:NoAdditionalReferencesAvailableFaultMessage"/>
    <fault name="NoReferencesAvailableFault"
        message="ogsi:NoReferencesAvailableFaultMessage"/>
    <fault name="NoSuchServiceFault"
        message="ogsi:NoSuchServiceFaultMessage"/>
    <fault name="NoSuchServiceStartedFault"
        message="ogsi:NoSuchServiceStartedFaultMessage"/>
    <fault name="ServiceHasTerminatedFault"
        message="ogsi:ServiceHasTerminatedFaultMessage"/>
    <fault name="TemporarilyUnavailableFault"
        message="ogsi:TemporarilyUnavailableFaultMessage"/>
    <fault name="RedirectionFault"
        message="ogsi:RedirectionFaultMessage"/>
    <fault name="Fault" message="ogsi:FaultMessage"/>
  </operation>
  <sd:serviceData name="handleResolverScheme"
                  type="xsd:anyURI"
                  minOccurs="1"
                  maxOccurs="unbounded"
                  mutability="mutable"
                  modifiable="false"
                  nillable="true"/>
</gwsdl:portType>

<!-- Factory PortType -->
<gwsdl:portType name="Factory" extends="ogsi:GridService">
  <operation name="createService">
    <input message="ogsi:CreateServiceInputMessage"/>
    <output message="ogsi:CreateServiceOutputMessage"/>
    <fault name="ExtensibilityNotSupportedFault"
        message="ogsi:ExtensibilityNotSupportedFaultMessage"/>
    <fault name="ExtensibilityTypeFault"
        message="ogsi:ExtensibilityTypeFaultMessage"/>
    <fault name="ServiceAlreadyExistsFault"
        message="ogsi:ServiceAlreadyExistsFaultMessage"/>
    <fault name="Fault" message="ogsi:FaultMessage"/>
```

```
  </operation>
  <sd:serviceData name="createServiceExtensibility"
                  type="ogsi:CreateServiceExtensibilityType"
                  minOccurs="1"
                  maxOccurs="unbounded"
                  mutability="static"
                  modifiable="false"
                  nillable="false"/>
</gwsdl:portType>

<!-- NotificationSource PortType -->
<gwsdl:portType name="NotificationSource" extends="ogsi:GridService">
  <operation name="subscribe">
    <input message="ogsi:SubscribeInputMessage"/>
    <output message="ogsi:SubscribeOutputMessage"/>
    <fault name="ExtensibilityNotSupportedFault"
        message="ogsi:ExtensibilityNotSupportedFaultMessage"/>
    <fault name="ExtensibilityTypeFault"
        message="ogsi:ExtensibilityTypeFaultMessage"/>
    <fault name="TargetInvalidFault"
        message="ogsi:TargetInvalidFaultMessage"/>
    <fault name="Fault" message="ogsi:FaultMessage"/>
  </operation>
  <sd:serviceData name="notifiableServiceDataName"
                  type="xsd:QName"
                  minOccurs="0"
                  maxOccurs="unbounded"
                  mutability="mutable"
                  modifiable="false"
                  nillable="false"/>
  <sd:serviceData name="subscribeExtensibility"
                  type="ogsi:OperationExtensibilityType"
                  minOccurs="1"
                  maxOccurs="unbounded"
                  mutability="static"
                  modifiability="false"
                  nillable="false"/>
  <sd:staticServiceDataValues>
    <ogsi:subscribeExtensibility
        inputElement="ogsi:subscribeByServiceDataNames"/>
  </sd:staticServiceDataValues>
</gwsdl:portType>

<!-- Notification Sink PortType -->
<gwsdl:portType name="NotificationSink">
  <operation name="deliverNotification">
    <input message="ogsi:DeliverNotificationInputMessage"/>
  </operation>
</gwsdl:portType>
```

```
<!-- NotificationSubscription PortType -->
<gwsdl:portType name="NotificationSubscription"
    extends="ogsi:GridService">
  <sd:serviceData name="subscriptionExpression"
                  type="xsd:anyType"
                  minOccurs="1"
                  maxOccurs="1"
                  mutability="mutable"
                  modifiable="false"
                  nillable="false"/>
  <sd:serviceData name="sinkLocator"
                  type="ogsi:LocatorType"
                  minOccurs="1"
                  maxOccurs="1"
                  mutability="mutable"
                  modifiable="false"
                  nillable="false"/>
</gwsdl:portType>

<!-- ServiceGroupEntry PortType -->
<gwsdl:portType name="ServiceGroupEntry" extends="ogsi:GridService">
  <sd:serviceData name="memberServiceLocator"
                  type="ogsi:LocatorType"
                  minOccurs="1"
                  maxOccurs="1"
                  mutability="mutable"
                  modifiable="false"
                  nillable="false"/>
  <sd:serviceData name="content"
                  type="ogsi:EntryContentType"
                  minOccurs="1"
                  maxOccurs="1"
                  mutability="mutable"
                  modifiable="false"
                  nillable="false"/>
</gwsdl:portType>

<!-- ServiceGroup PortType -->
<gwsdl:portType name="ServiceGroup" extends="ogsi:GridService">
        <sd:serviceData name="membershipContentRule"
                        type="ogsi:MembershipContentRuleType"
                        minOccurs="1"
                        maxOccurs="unbounded"
                        mutability="constant"
                        modifiable="false"
                        nillable="false"/>
        <sd:serviceData name="entry"
                        type="ogsi:EntryType"
```

```
                        minOccurs="0"
                        maxOccurs="unbounded"
                        mutability="mutable"
                        modifiable="false"
                        nillable="false"/>
</gwsdl:portType>

<!-- ServiceGroupRegistration PortType -->
<gwsdl:portType name="ServiceGroupRegistration"
    extends="ogsi:ServiceGroup">
  <operation name="add">
    <input message="ogsi:AddInputMessage"/>
    <output message="ogsi:AddOutputMessage"/>
    <fault name="ExtensibilityNotSupportedFault"
        message="ogsi:ExtensibilityNotSupportedFaultMessage"/>
    <fault name="ExtensibilityTypeFault"
        message="ogsi:ExtensibilityTypeFaultMessage"/>
    <fault name="ContentCreationFailedFault"
        message="ogsi:ContentCreationFailedFaultMessage"/>
    <fault name="UnsupportedMemberInterfaceFault"
        message="ogsi:UnsupportedMemberInterfaceFaultMessage"/>
    <fault name="AddRefusedFault"
        message="ogsi:AddRefusedFaultMessage"/>
    <fault name="Fault" message="ogsi:FaultMessage"/>
  </operation>
  <operation name="remove">
    <input message="ogsi:removeInputMessage"/>
    <output message="ogsi:removeOutputMessage"/>
    <fault name="ExtensibilityNotSupportedFault"
        message="ogsi:ExtensibilityNotSupportedFaultMessage"/>
    <fault name="ExtensibilityTypeFault"
        message="ogsi:ExtensibilityTypeFaultMessage"/>
    <fault name="MatchFailedFault"
        message="ogsi:MatchFailedFaultMessage"/>
    <fault name="RemoveFailedFault"
        message="ogsi:RemoveFailedFaultMessage"/>
    <fault name="Fault" message="ogsi:FaultMessage"/>
  </operation>
  <sd:serviceData name="addExtensibility"
                  type="ogsi:OperationExtensibilityType"
                  minOccurs="0"
                  maxOccurs="unbounded"
                  mutability="static"
                  modifiable="false"
                  nillable="false"/>
  <sd:serviceData name="removeExtensibility"
                  type="ogsi:OperationExtensibilityType"
                  minOccurs="1"
                  maxOccurs="unbounded"
```

```
                mutability="static"
                modifiable="false"
                nillable="false"/>
  <sd:staticServiceDataValues>
    <ogsi:removeExtensibility
        inputElement="ogsi:matchByLocatorEquivalence"/>
  </sd:staticServiceDataValues>
</gwsdl:portType>

</definitions>
```

19.2 *http://www.gridforum.org/namespaces/2003/03/serviceData*

```
<?xml version="1.0" encoding="UTF-8"?>
<schema
targetNamespace="http://www.gridforum.org/namespaces/2003/03/serviceData"
     xmlns:sd="http://www.gridforum.org/namespaces/2003/03/serviceData"
     xmlns="http://www.w3.org/2001/XMLSchema"
     elementFormDefault="qualified"
     attributeFormDefault="unqualified">

<attributeGroup name="occurs">
   <attribute name="minOccurs"
              type="nonNegativeInteger"
              use="optional"
              default="1"/>
   <attribute name="maxOccurs">
      <simpleType>
         <union memberTypes="nonNegativeInteger">
            <simpleType>
               <restriction base="NMTOKEN">
                  <enumeration value="unbounded"/>
               </restriction>
            </simpleType>
         </union>
      </simpleType>
   </attribute>
</attributeGroup>

<complexType name="ServiceDataType">
   <sequence>
      <any namespace="##any" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="name" type="NCName"/>
   <attribute name="type" type="QName"/>
   <attribute name="nillable"
              type="boolean"
              use="optional"
              default="false"/>
   <attributeGroup ref="sd:occurs"/>
```

```
   <attribute name="mutability" use="optional" default="extendable">
      <simpleType>
         <restriction base="string">
            <enumeration value="static"/>
            <enumeration value="constant"/>
            <enumeration value="extendable"/>
            <enumeration value="mutable"/>
         </restriction>
      </simpleType>
   </attribute>
   <attribute name="modifiable" type="boolean" default="false"/>
   <anyAttribute namespace="##other" processContents="lax"/>
</complexType>

<element name="serviceData" type="sd:ServiceDataType"/>

  <complexType name="ServiceDataValuesType">
    <sequence>
      <any namespace="##any" minOccurs="0" maxOccurs="unbounded" />
    </sequence>
  </complexType>

  <element name="serviceDataValues" type="sd:ServiceDataValuesType"/>
  <element name="staticServiceDataValues"
      type="sd:ServiceDataValuesType"/>
</schema>
```

19.3 *http://www.gridforum.org/namespaces/2003/03/gridWSDLExtensions*

```
<?xml version="1.0" encoding="UTF-8"?>
<schema
targetNamespace="http://www.gridforum.org/namespaces/2003/03/gridWSDLExtensions"
        xmlns:gwsdl="http://www.gridforum.org/namespaces/2003/03/gridWSDLExtensions"
        xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
        elementFormDefault="qualified">

  <import namespace="http://schemas.xmlsoap.org/wsdl/"/>

  <element name="portType" type="gwsdl:PortTypeType"/>
  <complexType name="PortTypeType">
    <complexContent>
      <extension base="wsdl:portTypeType">
        <sequence>
          <any namespace="##other" minOccurs="0"
              maxOccurs="unbounded"/>
        </sequence>
        <attribute name="extends" use="optional">
          <simpleType>
            <list itemType="QName"/>
```

```
          </simpleType>
        </attribute>
        <anyAttribute namespace="##other"/>
      </extension>
    </complexContent>
  </complexType>
</schema>
```