

GFD-R-P.056

ジョブサブミッション記述言語 (JSDL) 仕様

<http://forge.gridforum.org/projects/jsdl-wg>

著者 :

Ali Anjomshoaa, EPCC

Fred Brisard, CA

Michel Drescher, 富士通

Donal Fellows, UoM

An Ly, CA

Stephen McGough, LeSC

Darren Pulsipher, Ovoca LLC

Andreas Savva, 富士通 ( 編者 )

2005 年 11 月 7 日

## ジョブサブミッション記述言語 (JSDL) 仕様 バージョン 10

### 本文書の位置づけ

本文書は、ジョブサブミッション記述言語 (JSDL) の仕様に関し、グリッドコミュニティに情報を提供するものである。文書の配布に制限はない。

### 著作権に関する注意

グローバル・グリッド・フォーラム (2003-2005) がすべての著作権を有する。

### 概要

本文書では、ジョブサブミッション記述言語 (JSDL) のセマンティクスと構造について記述する。JSDL は、リソース、とくにグリッド環境にあるリソース (ただしリソースはこれに限らない) に計算ジョブをサブミットする際の要件を記述する際に使用する。JSDL の標準的な XML スキーマ、およびこのスキーマに基づいた JSDL の例も本文書で示す。

## 目次

### 概要

- 1 はじめに
- 2 表記法
- 3 JSDL の目的
  - 3.1 リソース要件言語 (RPL) について
  - 3.2 スケジューリング記述言語 (SDL) について
  - 3.3 ウェブサービスアグリメント (WS-AG) について
  - 3.4 ジョブポリシー言語 (JPL) について
  - 3.5 ジョブライフタイム管理言語 (JLML) について
  - 3.6 ワークフローについて
- 4 JSDL 文書構造
- 5 JSDL 要素タイプ
  - 5.1 規範的 XML スキーマタイプ
  - 5.2 JSDL タイプ
    - 5.2.1 ProcessorArchitectureEnumeration タイプ
    - 5.2.2 FileSystemTypeEnumeration タイプ
    - 5.2.3 OperatingSystemTypeEnumeration タイプ
    - 5.2.4 CreationFlagEnumeration タイプ
    - 5.2.5 RangeValue\_Type タイプ
- 6 JSDL コア要素セット
  - 6.1 ジョブ構造要素
    - 6.1.1 JobDefinition 要素
    - 6.1.2 JobDescription 要素
    - 6.1.3 Description 要素
  - 6.2 ジョブアイデンティティ要素
    - 6.2.1 JobIdentification 要素
    - 6.2.2 JobName 要素
    - 6.2.3 JobAnnotation 要素
    - 6.2.4 JobProject 要素
  - 6.3 アプリケーション要素
    - 6.3.1 Application 要素
    - 6.3.2 ApplicationName 要素
    - 6.3.3 ApplicationVersion 要素
  - 6.4 リソース要素

- 6.4.1 Resources 要素
- 6.4.2 CandidateHosts 要素
- 6.4.3 HostName 要素
- 6.4.4 FileSystem 要素
- 6.4.5 MountPoint 要素
- 6.4.6 MountSource 要素
- 6.4.7 DiskSpace 要素
- 6.4.8 FileSystemType 要素
- 6.4.9 ExclusiveExecution 要素
- 6.4.10 OperatingSystem 要素
- 6.4.11 OperatingSystemType 要素
- 6.4.12 OperatingSystemName 要素
- 6.4.13 OperatingSystemVersion 要素
- 6.4.14 CPUArchitecture 要素
- 6.4.15 CPUArchitectureName 要素
- 6.4.16 IndividualCPUSpeed 要素
- 6.4.17 IndividualCPUTime 要素
- 6.4.18 IndividualCPUCount 要素
- 6.4.19 IndividualNetworkBandwidth 要素
- 6.4.20 IndividualPhysicalMemory 要素
- 6.4.21 IndividualVirtualMemory 要素
- 6.4.22 IndividualDiskSpace 要素
- 6.4.23 TotalCPUTime 要素
- 6.4.24 TotalCPUCount 要素
- 6.4.25 TotalPhysicalMemory 要素
- 6.4.26 TotalVirtualMemory 要素
- 6.4.27 TotalDiskSpace 要素
- 6.4.28 TotalResourceCount 要素
- 6.4.29 追加リソース
- 6.5 データステージング要素
  - 6.5.1 DataStaging 要素
  - 6.5.2 FileName 要素
  - 6.5.3 FileSystemName 要素
  - 6.5.4 CreationFlag 要素
  - 6.5.5 DeleteOnTermination 要素
  - 6.5.6 Source 要素

- 6.5.7 URI 要素
- 6.5.8 Target 要素
- 7 JSDL の拡張
  - 7.1 属性拡張
  - 7.2 要素拡張
  - 7.3 JSDL "other"値のセマンティクス
- 8 規範的拡張
  - 8.1 POSIX 準拠ホスト上の実行ファイル
    - 8.1.1 POSIXApplication 要素
    - 8.1.2 Executable 要素
    - 8.1.3 Argument 要素
    - 8.1.4 Input 要素
    - 8.1.5 Output 要素
    - 8.1.6 Error 要素
    - 8.1.7 WorkingDirectory 要素
    - 8.1.8 Environment 要素
    - 8.1.9 WallTimeLimit 要素
    - 8.1.10 FileSizeLimit 要素
    - 8.1.11 CoreDumpLimit 要素
    - 8.1.12 DataSegmentLimit 要素
    - 8.1.13 LOckedMemoryLimit 要素
    - 8.1.14 MemoryLimit 要素
    - 8.1.15 OpenDescriptorsLimit 要素
    - 8.1.16 PipeSizeLimit 要素
    - 8.1.17 StackSizeLimit 要素
    - 8.1.18 CPUTimeLimit 要素
    - 8.1.19 ProcessCountLimit 要素
    - 8.1.20 VirtualMemoryLimit 要素
    - 8.1.21 ThreadCountLimit 要素
    - 8.1.22 UserName 要素
    - 8.1.23 GroupName 要素
- 9 セキュリティに関して
  - 著者情報
  - 寄稿者
  - 謝辞
  - 著作権情報

知的財産権について

規範的文献

参考となる文献

付録 1 JSDL 規範的スキーマ

付録 2 JSDL POSIX アプリケーション規範的スキーマ

付録 3 拡張した JSDL 参考例

## 1 はじめに

ジョブサブミッション記述言語 (JSDL) は、リソース、とくにグリッド環境にあるリソース (ただしリソースはこれに限らない) に計算ジョブをサブミットする際の要件を記述する言語である。JSDL は、これらの要件を XML の要素の集まりとして表現するためのボキャブラリーや標準 XML スキーマを含む。

JSDL の仕様は、大きく分けて 2 つのシナリオが動機となっている。1 つは、多くの組織にはさまざまなジョブ管理システムがあり、ジョブサブミッション要件を記述する独自の言語をシステムごとに持っている。これによりジョブ管理のためのシステム間相互運用性が難しくなっている。こうした異なるシステムをすべて活用するために、これらの多くの組織はいくつもの異なるジョブサブミッションに関するドキュメントを用意、維持しなければならない。ジョブサブミッションに関するこれらのドキュメントは、いずれも同じサブミッションを記述するものであるが、システムごとに作成しなければならない。JSDL のような標準言語はさまざまなシステムに簡単にマップすることができ、これを用いれば問題を解決できる。

2 つめのシナリオとして、グリッド環境では、異機種混在型の環境におけるいくつものさまざまなジョブ管理システムの間でやりとりが行われる点があげられる。図 1 はこうしたグリッド環境の一例を示している。グリッド上でのジョブサブミッションの記述 (図 1 の JSDL 文書として示されている) は、そのジョブを最初にサブミットした人が手に入れることのできなかつた情報により、さらに変更あるいは改良を受ける。この記述はシステム間で受け渡しを行うことができ、あるいはそのジョブに対するリソース要件に合致するリソース上でインスタンスを作成することもできる。こうしたやりとりは、各システム独自の言語に容易に翻訳できる 1 つの標準言語を使って、すべて自動的に行うことが可能である。

図 1 は、JSDL 文書をグリッド環境で使用するることのできる数少ないシステムを描いている。ただし JSDL 文書はこれらのシステムだけで使われているわけではなく、そのほかにも JSDL を使用するものは、会計システム、セキュリティシステム、アーカイブシステム、プロヴナンス (監査) システムなど、多く存在する。

JSDL 1.0 は、グリッド環境へのジョブサブミッションを記述するための重要なボキャブラリーを提供するものである。このボキャブラリーは、以下のような多くの既存のシステムによって特徴づけられている: Condor<sup>1</sup>、Globus Toolkit<sup>2</sup>、Load Sharing Facility (LSF)<sup>3</sup>、Portable Batch System (PBS)<sup>4</sup>、(Sun) GridEngine (SGE)<sup>5</sup>、Uniform Interface to Computing Resources (Unicore)<sup>6</sup>。また、次のような多くの既存の標準や提案されている標準によっても特徴づけられている: 分散型リソース管理アプリケーション API 仕様 1.0 [DRMAA]、共通情報モデル (CIM) 2.9.0 [CIM]、ポータブルオペレーティングシステムインタフェース (POSIX) 3 [POSIX]。

将来的には、非標準の拡張により、中心的な JSDL 1.0 仕様に対してより複雑な機能が提

供される。その機能のいくつかは、いずれ JSDL に対する正規の標準化された拡張となる可能性がある。

JSDL 1.0 の要素は、一般的に次のように分類される。

- ・ ジョブ同定要件
- ・ リソース要件
- ・ データ要件

-----  
<sup>1</sup>Condor プロジェクトのホームページ：

<sup>2</sup>Globus Toolkit のホームページ：

JSDL 1.0 の要素は、サブミット時のジョブ要件の記述のみに限定されている。したがってサブミット後のジョブに関する情報はまったく含まれていない。唯一のジョブ識別子やジョブステータス情報などは、基盤となるジョブ管理システムが通常管理しており、別のドキュメントから得られる場合や、拡張を通じて JSDL 1.0 準拠のドキュメントに追加される場合がある。第 3 節では、JSDL 1.0 の目的の背景にある理由付けについて解説する。

JSDL 1.0 言語要素は、第 4 節および第 6 節で定義する。規範的 XML スキーマについては付録 1 で与える。

JSDL 1.0 は、拡張性のある仕様である。第 7 節では、JSDL 1.0 の拡張性の仕組みについて解説し、第 8 節では規範的拡張について触れる。

WS Client	WS クライアント
WS Client	WS クライアント
WS Client	WS クライアント

WS Intermediary	中継 WS
-----------------	-------

Super-scheduler, or Broker, or...	スーパースケジューラ、ブローカなど
-----------------------------------	-------------------

Job Manager	ジョブマネージャ
-------------	----------

Local Information System	ローカル情報システム
--------------------------	------------

A Grid Information System	グリッド情報システム
---------------------------	------------

DRM	DRM
-----	-----

Local Resource	ローカルリソース
----------------	----------

Another JSDL System	他の JSDL システム
---------------------	--------------

## 図 1 グリッド環境における JSDL の使用

## 2 表記法

“MUST”、“MUST NOT”、“REQUIRED”、“SHALL”、“SHALL NOT”、“SHOULD”、“SHOULD NOT”、“RECOMMENDED”、“MAY”、“OPTIONAL”といったキーワードは、RFC-2119 [RFC-2119]に説明されたものとして解釈されるべきである。

各コンポーネントには、コンポーネントの記述に入る前に疑似スキーマが与えられている。疑似スキーマでは、属性や要素に対して BHF と同様の決まり事を用いている。たとえば“?”は 0 回または 1 回の発生、“\*”は 0 回以上の発生、“+”は 1 回以上の発生をそれぞれ表す。属性は通常、そのタイプに応じて規範的スキーマで定義されているように数値が割り当てられる。

本仕様では、ネームスペースプリフィクスを使用する。これらについては表 2 - 1 に一覧を示した。いずれのネームスペースプリフィクスも任意のものであり、セマンティックな重要性はない。

表 2-1: 本仕様書で用いるプリフィクスとネームスペース

プリフィクス	ネームスペース
xsd	<a href="http://www.w3.org/2001/XMLSchema">http://www.w3.org/2001/XMLSchema</a>
jsdl	<a href="http://schemas.ggf.org/jsdl/2005/11/jsdl">http://schemas.ggf.org/jsdl/2005/11/jsdl</a>
jsdl-posix	<a href="http://schemas.ggf.org/jsdl/2005/11/jsdl-posix">http://schemas.ggf.org/jsdl/2005/11/jsdl-posix</a>

JSDL 要素や JSDL 属性という言葉は、規範的 JSDL スキーマにおけるそれぞれの言語構築が XML 要素や XML 属性で表されることを意味している。

JSDL 文書という言葉は、付録 1 の規範的 JSDL スキーマの定義に照らし合わせて有効と判断される整った XML 文書を指す。

キーワードである present は、JSDL 要素を指すために使用する場合は、その要素のインスタンスが JSDL 文書に存在することを意味する。

support というキーワードは、JSDL 仕様と言語構築をサポートした JSDL 使用システムに関して言えば、JSDL 文書の構文解析を行うシステム上の機能を指している。すなわち、JSDL を使用したシステムは必ず言語構築を解釈でき、本仕様書に記載されたセマンティクスや JSDL 文書で与えられている値をその言語構築に対して指定することができなければならない。したがってすべての JSDL に準拠したシステムは、あらゆる JSDL 言語構築をサポートしている必要がある。

satisfy というキーワードは、JSDL 文書を満たしたシステムに関し、システムがその文書に存在するすべての要素をサポートし満足することができることを意味する。JSDL 使用システムが JSDL 文書内の要素に割り当てられた値を適切に実装することができる場合に、その要素は満たされたことになる。

### 3 JSDL の目的

JSDL は、それぞれのジョブのサブミッション要件を記述するための言語である。ジョブのライフサイクル全体やジョブ間の関係を扱うものではない。したがって JSDL を使って記述し、サブミットしたジョブを管理したり実行したりする際には、とくにそれがグリッド環境であれば、他の多くの言語やプロトコルも必要になる。たとえばワークフロー言語は JSDL を使って記述した個々のジョブの間にどのような関係があるのかを記述する際に必要になるものである。あるいはこれらのジョブとジョブが消費、生成するデータとの関係を記述する際にも使用する。他の例として、要件に基づいてジョブを最適なりソース環

境に投入する際に必要となるネゴシエーションについて考えてみる。この場合、ネゴシエーションとアグリメントの適切なプロトコルを使う必要がある。そうしたプロトコルの一例が、GGF WS-アグリメント仕様[WS-AG]で定義されたプロトコルである。図2では、とくにグリッドにおいてジョブをサブミット、管理、実行するために必要な他の要素のうちのいくつかを示している。これらの要素の多くは、まだ使用することのできない言語やプロトコルであり、これらは現在開発中か、あるいは規定し標準化を行う必要がある段階にある。

Workflow	ワークフロー
Job	ジョブ
JSDL	JSDL
RPL	RPL
SDL	SDL
WS-AG	WS-AG
JPL	JPL
JLML	JLML

図2 JSDL と他の仕様との関係

#### 凡例

RPL：リソース要件言語

SDL：スケジューリング記述言語

WS-AG：ウェブサービスアグリメント

JPL：ジョブポリシー言語

JLML：ジョブライフタイム管理言語

### 3.1 リソース要件言語（RPL）について

JSDL仕様は、ジョブに対するリソース要件を記述する言語を使用することができる。この言語をリソース要件言語（RPL）と呼ぶ。RPLの定義は、それ自体で完全な仕様となっているため、JSDLの範疇を超えている。適切なRPLがない場合は、RPLに含まれている要素のコアセットをわれわれが提供することになる。JSDLの将来の仕様では、RPLの標準的な仕様を得られるようになった場合にこれを使用することになるであろう。

### 3.2 スケジューリング記述言語（SDL）について

ジョブのスケジューリングやスケジューリング要件の記述は、複雑な作業となる。スケジューリング要件は、たとえば以下のようないくつかのカテゴリに分類することができる。

- ・一時的スケジューリング：ジョブの開始時間と終了時間、ジョブ実行のための有効性画面など。
- ・データ依存スケジューリング：入力データの予想されるアベイラビリティや有効性に基づいてジョブをスケジューリングする。
- ・ワークフロー依存スケジューリング：ワークフローにおけるジョブの順番にしたがってジョブをスケジューリングする。

さらに、1つのジョブに対する複数のスケジューリング要件に関し、これらのカテゴリ間に何らかの依存関係が存在する場合がある。たとえば、ワークフローで定義されたジョブの順番と、スケジュールされたジョブに必要な入力データとの間に、依存関係があるかもしれないのである。

ジョブのスケジューリング要件を記述するのは複雑な作業になるため、スケジューリング要件は JSDL の対象外とする判断がなされてきた。しかしジョブをリソースにサブミットし、グリッドのような潜在的に複雑な環境でこれらのジョブを管理、実行するためのスケジューリング記述言語 (SDL) については、その重要性をわれわれは十分認識している。

### 3.3 ウェブサービスアグリメント (WS-AG) について

リソースのウェブサービス・フロントエンドとジョブサブミッション・ウェブサービスとの間のアグリメントの構築は、現在のところ、GGF WS-アグリメント (WS-AG) 仕様 [WS-AG] 範疇に入る。WS-アグリメントは、ジョブをリソースにサブミットするための重要な新しい標準である。しかし現在使用されている多くのシステムにはウェブサービス・フロントエンドがなく、その機能を必要としていない。JSDL が他の標準とともに使用できることは重要であるが、JSDL を独立に使用できるということもやはり重要である。つまり、ジョブサブミッションの JSDL 記述は、WS-AG の有無にかかわらず、デプロイすることができ、使用することができるのである。(なお WS-AG は、OASIS WSRF 技術委員会<sup>7</sup>が開発している仕様のウェブサービス・リソースフレームワーク (WSRF) セットに基づいている。)

### 3.4 ジョブポリシー言語 (JPL) について

ジョブポリシーは非常に幅の広いテーマであり、ジョブ実行のさまざまな側面に対してポリシーを適用することができる。ユーザは自分のジョブにポリシーを適用したいと考え、リソースの所有者は自分のリソースを使って独自のポリシーをジョブに適用したいと考えるかもしれない。こうしたポリシーの制限を表現するには特別な言語が必要であるため、そのポリシーの定義は JSDL の範囲外となっている。ジョブポリシー言語 (JPL) は JSDL の姉妹言語と見なすことができる。

### 3.5 ジョブライフタイム管理言語 (JLML) について

ジョブのライフタイム管理は、ジョブ管理全体の重要な部分である。ライフタイム管理により、ジョブやジョブ制御指令の状態が記述可能になる。このためジョブ管理のこうした側面を記述することのできるジョブライフタイム管理 (JLM) 言語が必要となるのである。ライフタイム管理はサブミット後のジョブに関するものなので、JLM は JSDL の範囲外となっている。

### 3.6 ワークフローについて

JSDL は、個々のジョブをリソースにサブミットするためのジョブテンプレートを記述するために設計された。しかしパラメトリックジョブ、コアロケーション、ワークフローなどを記述する言語の開発にも大きな関心があるとわれわれは認識している。こうした言語は JSDL を意識したものである。すなわちこれらの開発活動の中で、各ノードのジョブに関する JSDL 記述をこれらの言語が参照していることもある。しかしわれわれは、このような言語の定義は JSDL のような標準を基礎に構築すべき別の独立した作業であると考えており、したがって JSDL の範疇には入っていない。

## 4 JSDL 文書構造

JSDL 文書は XML を使って記述されており、付録 1 にある規範的 XML スキーマに基づいている。

JSDL 文書は次のように構成されている。ルート要素である `JobDefinition` には、`JobDescription` という必須の子要素が 1 つある。`JobDescription` にはジョブを記述する要素である `JobIdentification`、`Application`、`Resources`、`DataStaging` が含まれている。疑似スキーマの定義は次のように書かれる。

-----

<sup>7</sup>OASIS ウェブサービス・リソースフレームワーク TC

JSDL 文書全体の例が付録 3 にある。

JSDL 仕様では、JSDL 文書にない要素に対してデフォルト値を定義することはない。(通常、こうした要素の値は、JSDL を使用するシステムにまかせられる。詳しくは各要素の仕様を参照のこと。) 特定のジョブサブミッションに必要な JSDL 要素は、適切な値とともに JSDL 文書内に存在しているものと想定している。

文書全体が満たされるようためには、JSDL 文書内にあるすべての要素が満足されなければならない。もちろん満足することのできない JSDL 文書を作成することは可能である。たとえば相反するリソース要件を含む JSDL 文書は満足されない。

ただし、JSDL 文書を使用システムにサブミットした結果については、JSDL 仕様の範囲外である。

## 5 JSDL 要素タイプ

JSDL 仕様では、多くの規範的 XML スキーマタイプを使用しており、ジョブ要件の記述に特有の多くのタイプについても定義している。

JSDL-WG は、多くのタイプの定義を、既存の標準とくに共通情報モデル (CIM) やポータブルオペレーティングシステムインタフェース (POSIX) に基づいて作成している。これらのタイプについて規範的 XML スキーマの定義がないため、本仕様書で与えている。オペレーティングシステムタイプ (5.2.3 節参照) は CIM[CIM] をもとに定義されている。

多くのタイプが特別の値 "other" を定義している。この値を使って、他の仕様から要素を導入することができる。詳しくは 7.3 節を参照のこと。

### 5.1 規範的 XML スキーマタイプ

JSDL 仕様では、次のような規範的 XML スキーマ (xsd) タイプを採用している。

表 5-1 規範的 XML スキーマタイプ

XSD types      xsd タイプ

### 5.2 JSDL タイプ

JSDL 1.0 では次のタイプが定義されている。ProcessorArchitectureEnumeration、FileSystemTypeEnumeration、OperatingSystemTypeEnumeration、CreationFlagEnumeration、RangeValue\_Type。

#### 5.2.1 ProcessorArchitectureEnumeration タイプ

プロセッサアーキテクチャ・エニユメレーションは、少数の共通プロセッサアーキテク

チャのインストラクションセットアーキテクチャ (ISA) 名に基づいている。サポートされていない値を以下に列記する。

表 5-2 プロセッサアーキテクチャ (jsdl: ProcessorArchitectureEnumeration)

規範的 JSDL 名	定義
sparc	SPARC アーキテクチャプロセッサ
powerpc	PowerPC アーキテクチャプロセッサ
x86	8086 チップセットから派生した Intel アーキテクチャプロセッサ
X86_32	32 ビット処理が可能な x86 プロセッサ
X86_64	64 ビット処理が可能な x86 プロセッサ
parisc	PARIS アーキテクチャプロセッサ C
mips	MIPS アーキテクチャプロセッサ
ia64	Intel アーキテクチャ 64 ビットプロセッサ
arm	ARM プロセッサ
other	このエニユメレーションでは値が定義されていない

このタイプのリストはさらに拡張することができる。このトークンの拡張は JSDL ネームスペースに入れてはいけない。

### 5.2.2 FileSystemTypeEnumeration タイプ

サポートされていないファイルシステムタイプを以下に示す。

表 5-3 ファイルシステムタイプ (jsdl: FileSystemTypeEnumeration)

規範的 JSDL 名	定義
swap	メモリをページアウトするための通常スワップスペース
temporary	定期的に削除するファイルのための通常テンポラリスペース。このスペースはジョブが完了した後では使用できない。
spool	ジョブが完了したあとでも残っている可能性があるファイルのためのテンポラリスペース
normal	書き込みや読み出しをするファイルのための標準スペース。ファイルは、ユーザが明確に要求しないかぎり、ジョブが終わっても削

	除されない。
--	--------

このタイプのリストはさらに拡張することができる。このトークンの拡張は JSDL ネームスペースに入れてはいけない。

### 5.2.3 OperatingSystemTypeEnumeration タイプ

以下の値は、JSDL を使用するシステムでサポートされていなければならない。この値は、CIM\_OperatingSystem モデル[CIM]の OSType のフィールドに基づいている。

表 5-4 オペレーティングシステムタイプ (jsdl: OperatingSystemTypeEnumeration )

Normative JSDL Names                      規範的 JSDL 名

このタイプのリストはさらに拡張することができる。このトークンの拡張は JSDL ネームスペースに入れてはいけない。

### 5.2.4 CreationFlagEnumeration タイプ

サポートしなければならないファイル作成フラグは以下の通りである。

表 5-5 ファイル作成フラグ (jsdl: CreationFlagEnumeration )

規範的 JSDL 名	定義
overwrite	同じ名前を持った既存のファイルを上書きする。あるいは新しいファイルを作成する。
dontOverwrite	同じ名前を持った既存のファイルを上書きしない。
append	同じ名前を持った既存のファイルに追加する。あるいは新しいファイルを作成する。

### 5.2.5 RangeValue\_Type タイプ

レンジ値は、( オプションで"epsilon"引数をつけた ) 正確な値、左開区間、右開区間、範囲の定義が可能な複合型である。与えられたすべての数値は xsd:double のタイプである。UpperBoundedRanges と LowerBoundedRanges はそれぞれ上限と下限に限られる。レンジは、JSDL を使用するシステムの性能によって、負の無限大あるいは正の無限大まで"unlimited"でもよい。たとえば Java で表現した場合、"infinity" ( 無限大 ) の実装はそれぞれ

java.lang.Double.NEGATIVE\_INFINITY と java.lang.Double.POSITIVE\_INFINITY

となる。

オプションの属性"exclusiveBound"は、とくに指定しない限り、デフォルト値である"false"となっている。オプションの属性"epsilon"が指定されていない場合は、デフォルト値である0になっており、JSDLを使用するシステムは正確な一致を行わなければならない。

交差するレンジを指定する RangeValues は、与えられたジョブの記述に合致するよう、JSDL 使用システムによってコラプスさせられることがあるが、JSDL 文書は変更してはならない。

RangeValue\_Type タイプは、JSDL を使用するシステムがサポートしていなければならない。正規のマッチングセマンティクスは次節(5.2.5.1)で定義する。

#### 5.2.5.1 マッチングセマンティクス

次のマッチングセマンティクスを使用しなければならない。

・以下の条件のうち、少なくとも1つが満たされている場合に限り、数値 N が RangeValue の R に一致する。

R が UpperBoundedRange の U を含み、exclusiveBound 属性は偽、そして  $N = U$  である。

R が UpperBoundedRange の U を含み、exclusiveBound 属性は真、そして  $N < U$  である。

R が LowerBoundedRange の L を含み、exclusiveBound 属性は偽、そして  $N = L$  である。

R が LowerBoundedRange の L を含み、exclusiveBound 属性は真、そして  $N > L$  である。

R が Exact の E を含み、epsilon 属性 e が  $E - e \leq N \leq E + e$  を満たす。

R が LowerBoundRange の L と UpperBoundRange の U を含み、以下の2点とも真である。

・L の exclusiveBound 属性は偽であり  $N = L$  である。あるいはL の exclusiveBound 属性は真であり  $N > L$  である。

・U の exclusiveBound 属性は偽であり  $N = U$  である。あるいはU の exclusiveBound 属性は真であり  $N < U$  である。

#### 5.2.5.2 疑似スキーマ

##### 5.2.5.3 例

疑似表現"5, 6.7777, 7.0, [50.3,99.5), [100-" (これは、5、6.7777、7.0の数値から構成さ

れる disjoint range を意味する。すべての数値は 50.3 以上 99.5 未満の値を含んでおり、すべての値は 100 以上である。) は、RangeValue に以下のようにエンコードすることができる。

## 6 JSDL コア要素セット

JSDL コア要素セットには、JSDL 1.0 で定義される要素に対するセマンティクスが含まれる。すべての要素は JSDL 1.0 に準拠したシステムでサポートされていなければならない。(第 2 節も参照のこと。)

### 6.1 ジョブ構造要素

#### 6.1.1 JobDefinition 要素

##### 6.1.1.1 定義

この要素はジョブと要件を記述するものである。JobDescription のセクションを含む。この要素は JDSL 文書のルート要素である。

##### 6.1.1.2 多重度

この要素の多重度は 1 である。

##### 6.1.1.3 タイプ

この要素は複合型であり、次の要素をサポートしていなければならない。

- JobDescription

##### 6.1.1.4 属性

以下の属性が定義されている。

- id: ジョブ定義文書の id。これは"xsd:ID"として定義され、文書のデフォルトのネームスペースにある。id は省略してもよい。

##### 6.1.1.5 疑似スキーマ

##### 6.1.1.6 例

“gnuplot”という id のある文書

#### 6.1.2 JobDescription 要素

##### 6.1.2.1 定義

この要素はジョブと要件を記述するものである。JobIdentification、Application、Resources、DataStaging の要素を含む。

##### 6.1.2.2 多重度

この要素の多重度は 1 である。

##### 6.1.2.3 タイプ

この要素は複合型であり、次の要素をサポートしていなければならない。

- JobIdentification

- Application

- ・ Resources
- ・ DataStaging

#### 6.1.2.4 属性

属性は定義されていない。

#### 6.1.2.5 疑似スキーマ

### 6.1.3 Description 要素

#### 6.1.3.1 定義

この要素は、これが含む複合型要素に関して、人が読める形で記述情報を提供するものである。JobIdentificatoin、Application、FileSystem など、他の多くの JSDL 要素の下位要素として存在することがある。この要素が下位要素として存在していない場合は、記述は定義されない。

#### 6.1.3.2 多重度

この要素の多重度は 0 または 1 である。

#### 6.1.3.3 タイプ

この要素のタイプは xsd:string である。

#### 6.1.3.4 属性

属性は定義されていない。

#### 6.1.3.5 疑似スキーマ

### 6.2 ジョブアイデンティティ要素

#### 6.2.1 JobIdentification 要素

##### 6.2.1.1 定義

この要素は、JobName、Descripton、JobAnnotation、JobProject など、ジョブを特定するすべての要素を含んでいる。この要素が存在しない場合は、その値やあらゆる下位要素がいずれも定義されない。

##### 6.2.1.2 多重度

この要素の多重度は 0 または 1 である。

##### 6.2.1.3 タイプ

この要素は複合型であり、以下の要素をサポートしていなければならない。

- ・ JobName
- ・ Descripton
- ・ JobAnnotation
- ・ JobProject

##### 6.2.1.4 属性

属性は定義されていない。

##### 6.2.1.5 疑似スキーマ

## 6.2.2 JobName 要素

### 6.2.2.1 定義

この要素は、JSDL 文書で示されたジョブに名前を付けるため、ユーザが定めることができる文字列である。この要素は特定の JSDL 文書に固有のものではなく、ユーザが複数の JSDL 文書に対して同じ JobName (ジョブ名) をつけてもよい。この要素が存在しない場合はジョブ名は定義されない。

P.17

### 6.2.2.2 多重度

この要素の多重度は 0 または 1 である。

### 6.2.2.3 タイプ

この要素のタイプは xsd:string である

### 6.2.2.4 属性

属性は定義されていない。

### 6.2.2.5 疑似スキーマ

### 6.2.2.6 例

ジョブ名は、検索やソートあるいはジョブ管理といった目的に使用することができる。

## 6.2.3 JobAnnotation 要素

### 6.2.3.1 定義

この要素は、ジョブに注釈を付けるためにユーザが与えることができる文字列である。この要素が存在しない場合、その文字列は定義されない。Description 要素と違い、JobAnnotation 要素は JSDL 使用システムが使用することになっている情報を含んでいてもよい。

### 6.2.3.2 多重度

この要素の多重度は 0 または 1 である。

### 6.2.3.3 タイプ

この要素のタイプは xsd:string である。

### 6.2.3.4 属性

属性は定義されていない。

### 6.2.3.5 疑似スキーマ

### 6.2.3.6 例

注釈を使うことで、JSDL を使用しているシステムにとって意味のある固有の識別子を与えることができる。

## 6.2.4 JobProject 要素

### 6.2.4.1 定義

この要素は、ジョブが属するプロジェクトを指定する文字列である。プロジェクトは会計システムやアクセス制御システムで使用することができる。JobProject 要素の解釈は、JSDL 使用システムの実装にまかせている。この要素が存在しない場合は、定義されることはない。

#### 6.2.4.2 多重度

この要素の多重度は 0 または 1 である。

#### 6.2.4.3 タイプ

この要素のタイプは xsd:string である。

#### 6.2.4.4 属性

属性は定義されていない。

#### 6.2.4.5 疑似スキーマ

#### 6.2.4.6 例

ジョブがプロジェクトグループ"wcdma"に属しているものとする。

### 6.3 アプリケーション要素

JSDL 1.0 では、個々のジョブの記述のみをサポートしている。(詳しくは 3 章を参照のこと。) 1 つの JSDL 文書で定義できるアプリケーションは 1 つまでである。

#### 6.3.1 Application 要素

##### 6.3.1.1 定義

この要素はアプリケーションとその要件を記述するものであり、ApplicationName、ApplicationVersion、Description の各要素を含んでいる。この要素は、より明確なアプリケーションの定義を入れるためのハイレベルな一般的コンテナとしての役割を持っている。そうした定義の 1 つが、8.1.1 節で与えられる POSIX 準拠の規範的拡張の定義である。拡張なしにこの要素を使用した場合、アプリケーションを名前とバージョン番号で統一的に記述する。この要素が存在しない場合は、このジョブ定義が実行すべきアプリケーションを定義しないことになる。JSDL 文書は、データステージングのジョブやヌルジョブを定義することもできる。6.3.2 節および 8.1.2 節を参照のこと。

##### 6.3.1.2 多重度

この要素の多重度は 0 または 1 である。

##### 6.3.1.3 タイプ

この要素は複合型であり、以下の要素をサポートしていなければならない。

- ApplicationName
- ApplicationVersion
- Description

##### 6.3.1.4 属性

属性は定義されていない。

##### 6.3.1.5 疑似スキーマ

## 6.3.2 ApplicationName 要素

### 6.3.2.1 定義

この要素はアプリケーションの名前を定義する文字列である。これを使うことにより、ホストやシステムにおけるアプリケーションの実行ファイルの場所には関係なく、アプリケーションを特定することができる。この要素が存在しない場合は定義されることもない。さらに、実行するアプリケーションを定義するアプリケーション拡張要素が存在しなければ、ヌルジョブとなる。6.3.1 節および 8.1.2 節も参照のこと。

### 6.3.2.2 多重度

この要素の多重度は 0 または 1 である。

### 6.3.2.3 タイプ

この要素のタイプは `xsd:string` である。

### 6.3.2.4 属性

属性は定義されていない。

### 6.3.2.5 擬似スキーマ

### 6.3.2.6 例

BLAST アプリケーションの場合

## 6.3.3 ApplicationVersion 要素

### 6.3.3.1 定義

この要素は実行するアプリケーションのバージョンを定義する文字列である。JSDL 使用システムでは、アプリケーションのバージョンを選ぶ際に、厳密なテキストの一致が必要になる。この要素が存在しない場合は定義が行われず、いかなるバージョンのアプリケーションでも実行することができる。

### 6.3.3.2 多重度

この要素の多重度は 0 または 1 である。

### 6.3.3.3 タイプ

この要素のタイプは `xsd:string` である。

### 6.3.3.4 属性

属性は定義されていない。

### 6.3.3.5 擬似スキーマ

### 6.3.3.6 例

BLAST アプリケーション・バージョン 1.4 の場合

## 6.4 リソース要素

リソース要素はジョブのリソース要件を記述するものである。使用できるリソース要素は 1 つまでである。3.1 節で議論したように、リソース要素は要素のコアセットのみをサポ

ートしている。一様なリソースの記述をおもに目的としている。リソース要件の追加は拡張の形で定義できる。

#### 6.4.1 Resources 要素

##### 6.4.1.1 定義

この要素はジョブのリソース要件を含むものである。この要素が存在しない場合には、JSDL を使用するシステムはジョブを実行するためにどのようなリソースでも選ぶことが可能になる。

JSDL 文書の Resources 要素の中には、下に列記した Resources の下位要素の組み合わせが存在していることがある。とくに同じタイプあるいは異なるタイプの”Individual”や”Total”の要素が Resources 要素に含まれることがある。ただし文書全体が満たされるためには、JSDL 文書に存在するすべての要素が満足されなければならない。(4 章も参照のこと。)

##### 6.4.1.2 多重度

この要素の多重度は 0 または 1 である。

##### 6.4.1.3 タイプ

この要素は複合型であり、以下の要素をサポートしていなければならない。

##### 6.4.1.4 属性

属性は定義されていない。

##### 6.4.1.5 擬似スキーマ

#### 6.4.2 CandidateHosts 要素

##### 6.4.2.1 定義

この要素は複合型であり、名前をつけられたホストのうちジョブを実行するために選ばれる可能性のあるものを指定する際に使用する。この要素が存在していれば、ジョブを実行するためには、このようなホスト候補から 1 つあるいは複数のホストを選ばなければならない。この要素が存在しない場合は定義が行われることもない。名前をつけられたホストとしては、単独のホスト(たとえばマシン名)、ホストの論理グループ(たとえば名前をつけられた論理グループあるいはクラスタ)、仮想マシン、などがある。

##### 6.4.2.2 多重度

この要素の多重度は 0 または 1 である。

##### 6.4.2.3 タイプ

この要素は複合型であり、以下の要素をサポートしていなければならない。

- ・ HostName

##### 6.4.2.4 属性

属性は定義されていない。

##### 6.4.2.5 擬似スキーマ

##### 6.4.2.6 例

“bach.exampke.com”というマシンを使用する場合。

使用する可能性のあるマシンをリストアップする場合。

これらのマシンは、その一部あるいはすべてを使用することができる。

#### 6.4.3 HostName 要素

##### 6.4.3.1 定義

この要素は単純型であり、あるホストの名前 1 つを含んでいる。その名前は、ある 1 つのホスト（たとえばホスト名）、複数のホストからなる 1 つの論理グループ（たとえば名前をつけられた論理グループあるいはクラスタ）、仮想マシンなどを指している。

##### 6.4.3.2 多重度

この要素の多重度は 1 以上である。

##### 6.4.3.3 タイプ

この要素のタイプは `xsd:string` である。

##### 6.4.3.4 属性

属性は定義されていない。

##### 6.4.3.5 擬似スキーマ

##### 6.4.3.6 例

“bach.example.com”というマシンの場合。

“the-composers”という名の論理グループの場合。

#### 6.4.4 FileSystem 要素

##### 6.4.4.1 定義

この要素はジョブが必要とするファイルシステムを記述するものである。この要素は複合型であり、ファイルシステムを使用できる場所、必要なディスクスペースの大きさ、ファイルシステムの種類などを含んでいる。ファイルシステムはリソース上（たとえばローカルディスク上）のローカルなものである場合と、リモート（たとえば NFS マウント）に置かれる場合とがある。

##### 6.4.4.2 多重度

この要素の多重度は 0 以上である。

##### 6.4.4.3 タイプ

この要素は複合型であり、以下の要素をサポートしていなければならない。

- Description
- MountPoint
- MountSource
- DispSpace
- FileSystemType

##### 6.4.4.4 属性

以下の属性が定義されている。

・ name: ファイルシステムの名前。タイプは xsd:NCName である。この名前は、ユーザが定義するものであり、JSDL 文書内で固有のものでなければならない。広くサポートされて名前がよく知られているファイルシステムについて、その名前を定義しここで使用することができる。詳しくは 6.4.4.6 節を参照のこと。

#### 6.4.4.5 擬似スキーマ

#### 6.4.4.6 よく知られたファイルシステム名

JSDL を使用するシステムは通常、少数のよく知られた FileSystem をサポートするものと想定されている。これらのよく知られた FileSystem の名前とセマンティクスを、最小限の定義の例とともにここにリストアップする。これらの FileSystem はデフォルトでは使用できず、ジョブの実行に必要な場合は JSDL 文書内で定義しなければならない。

例として取り上げる宣言は、実際のサブミッションのためには、FileSystem のマウントポイントや使用可能なスペースなどを定義することで、より特化させることができる。

“HOME” FileSystem:

“ROOT” FileSystem:

“SCRATCH” FileSystem:

“TMP” FileSystem:

#### 6.4.4.7 例

FileSystem “HOME”の容量は 10 ギガバイト以下でなければならない。マウントポイントについては値が与えられていないので、実行システムが適切なマウントポイントを選ぶことができる。

FileSystem “HOME”の容量は 1 ギガバイト以上でなければならない。また”HOME”は MountPoint に書かれた場所に与えられる必要がある。

FileSystem “TMP” の容量は 1 ギガバイト以上でなければならない。実行システムは適切なマウントポイントを選ぶことができる。

FileSystem “FASTSTORAGE”が使用できなければならない。実行システムは適切なマウントポイントを選ぶことができる。拡張により、高性能などの更なる要件が指定される。

### 6.4.5 MountPoint 要素

#### 6.4.5.1 定義

この要素は、ジョブにアロケートされたリソース上で使用できるローカルな場所を記述する文字列である。

MountPoint が定義されていない場合、必要な FileSystem を提供するローカルな場所を JSDL 使用システムが選ばなければならない。

#### 6.4.5.2 多重度

この要素の多重度は 0 または 1 である。

#### 6.4.5.3 タイプ

この要素のタイプは `xsd:string` である。

#### 6.4.5.4 属性

属性は定義されていない。

#### 6.4.5.5 擬似スキーマ

#### 6.4.5.6 例

FileSystem “HOME”が”/home/darren”になければならない場合。

FileSystem “HOME”が”c: Documents and Setting fred”になければならない場合。

POSIX アプリケーションに対し、MountPoint 要素を定義せずに FileSystem “HOME”がリクエストされた。マウントポイントの実際の値を保持するために環境変数”HOME”が定義される。

この FileSystem がローカルな場所”/usr/x00a10”にあるとする。このジョブの環境には次の環境変数が含まれることになる。

### 6.4.6 MountSource 要素

#### 6.4.6.1 定義

この要素は、ジョブがローカルに使用できるリモートロケーションを記述する文字列である。

#### 6.4.6.2 多重度

この要素の多重度は 0 または 1 である。

#### 6.4.6.3 タイプ

この要素のタイプは `xsd:string` である。

#### 6.4.6.4 属性

属性は定義されていない。

#### 6.4.6.5 擬似スキーマ

#### 6.4.6.6. 例

FileSystem “HOME の容量は 10 ギガバイト以上でなければならない。マウントポイントについては値が与えられていないので、実行システムが適切なマウントポイントを選ぶことができる。そこで NFS エクスポート”sputnick.ggf.org:/home/steve”をマウントする必要がある。

### 6.4.7 DiskSpace 要素

#### 6.4.7.1 定義

この要素は、記載されている FileSystem 要素上でジョブが必要とするディスクスペースを与えるレンジ値である。ディスクスペースの大きさはバイトを単位に与えられる。この要素が存在しない場合はディスクスペースの大きさが定義されることはなく、JSDL を使用するシステムがどのような値でも選ぶことができる。

#### 6.4.7.2 多重度

この要素の多重度は 0 または 1 である。

#### 6.4.7.3 タイプ

この要素のタイプは `jsdl:RangeValue_Type` である。

#### 6.4.7.4 属性

属性は定義されていない。

#### 6.4.7.5 擬似スキーマ

#### 6.4.7.6. 例

少なくとも 1 ギガバイトのフリースペースを持った FileSystem の場合。

1 ギガバイト以下のフリースペースを持った FileSystem の場合。

### 6.4.8 FileSystemType 要素

#### 6.4.8.1 定義

この要素は、含まれている FileSystem 要素のファイルシステムのタイプを記述するトークンである。この要素が存在しない場合はファイルシステムのタイプが定義されることはなく、JSDL 使用システムがどのような値でも選ぶことができる。

#### 6.4.8.2 多重度

この要素の多重度は 0 または 1 である。

#### 6.4.8.3 タイプ

この要素のタイプは `jsdl:FileSystemTypeEnumeration` である。

#### 6.4.8.4 属性

属性は定義されていない。

#### 6.4.8.5 擬似スキーマ

#### 6.4.8.6. 例

### 6.4.9 ExclusiveExecution 要素

#### 6.4.9.1 定義

この要素は、JSDL 使用システムによりアロケートされたリソースに対してジョブが排他的アクセスを行うようにするかどうかを指定する論理型である。この要素が存在しない場合、定義も行われず、システムがどのような値でも選ぶことができる。

#### 6.4.9.2 多重度

この要素の多重度は 0 または 1 である。

#### 6.4.9.3 タイプ

この要素のタイプは `xsd:boolean` である。

- ・ 真：そのリソース上で排他的にジョブが実行される。
- ・ 偽：他のジョブが同時に実行される。

#### 6.4.9.4 属性

属性は定義されていない。

#### 6.4.9.5 擬似スキーマ

#### 6.4.9.6. 例

排他的実行の場合。

### 6.4.10 OperatingSystem 要素

#### 6.4.10.1 定義

この要素は複合型であり、ジョブが必要とするオペレーティングシステムを定義するものである。ここには `Description`、`OperatingSystemVersion`、`OperatingSystemType` の要素が含まれている。`OperatingSystem` 要素が存在しない場合には定義が行われず、JSDL を使用するシステムがどのような値でも選ぶことができる。

#### 6.4.10.2 多重度

この要素の多重度は 0 または 1 である。

#### 6.4.10.3 タイプ

この要素は複合型であり、以下の要素をサポートしていなければならない。

- ・ `OperatingSystemType`
- ・ `OperatingSystemVersion`
- ・ `Description`

#### 6.4.10.4 属性

属性は定義されていない。

#### 6.4.10.5 擬似スキーマ

### 6.4.11 OperatingSystemType 要素

#### 6.4.11.1 定義

この要素は複合型であり、オペレーティングシステムの名前を含むものである。この要素が存在しない場合には定義が行われず、JSDL を使用するシステムがどのような値でも選ぶことができる。JSDL `OperatingSystemTypeEnumeration` ( 5.2.3 節参照 ) が定義しない値は、特別なトークン "other" を指定し、拡張としてその値を含めることによって使用することができる ( 7.3 節参照 )。以下の例を参照のこと。

#### 6.4.11.2 多重度

この要素の多重度は 0 または 1 である。

#### 6.4.11.3 タイプ

この要素は複合型であり、以下の要素をサポートしていなければならない。

・ OperatingSystemName

#### 6.4.11.4 属性

属性は定義されていない。

#### 6.4.11.5 擬似スキーマ

#### 6.4.11.6 例

Inferno OS の場合。

Windows2003 サーバ OS 場合 ( JSDL の拡張メカニズム"other"を使用している )

### 6.4.12 OperatingSystemName 要素

#### 6.4.12.1 定義

この要素はオペレーティングシステムの名前を含んだトークンタイプである。

#### 6.4.12.2 多重度

この要素の多重度は 1 である。

#### 6.4.12.3 タイプ

この要素のタイプは jsdl:OperatingSystemTypeEnumeration である。

#### 6.4.12.4 属性

属性は定義されていない。

#### 6.4.12.5 擬似スキーマ

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

#### 6.4.12.6 例

Inferno OS の場合。

### 6.4.13 OperatingSystemVersion 要素

#### 6.4.13.1 定義

この要素はジョブが必要とするオペレーティングシステムのバージョンを定義する文字列である。JSDL を使用するシステムは、そのオペレーティングシステムのバージョンを選ぶ際に厳密なテキストの一致を必要とする。この要素が存在しない場合は、いかなるバージョンのオペレーティングシステムでも使用することができる。

#### 6.4.13.2 多重度

この要素の多重度は 0 または 1 である。

#### 6.4.13.3 タイプ

この要素のタイプは xsd:string である。

#### 6.4.13.4 属性

属性は定義されていない。

#### 6.4.13.5 擬似スキーマ

#### 6.4.13.6 例

バージョン番号”5.01”のオペレーティングシステムの場合。

#### 6.4.14 CPUArchitecture 要素

##### 6.4.14.1 定義

この要素は、実行環境にあるジョブが必要とする CPU アーキテクチャを指定する文字列である。この要素が存在しない場合は定義が行われず、JSDL を使用するシステムはいかなる値でも選ぶことができる。JSDL ProcessorArchitectureEnumeration ( 5.2.1 節参照 ) が定義しない値は、特別なトークン”other”を指定し、拡張としてその値を含めることによって使用することができる ( 7.3 節参照 )。以下の例を参照のこと。

##### 6.4.14.2 多重度

この要素の多重度は 0 または 1 である。

##### 6.4.14.3 タイプ

この要素は複合型であり以下の要素をサポートしなければならない。

- ・ CPUArchitectureName

##### 6.4.14.4 属性

属性は定義されていない。

##### 6.4.14.5 擬似スキーマ

##### 6.4.14.6 例

SPARC アーキテクチャマシンの場合。

CELL アーキテクチャマシンの場合 ( JSDL の拡張メカニズム”other”を使用している )。

#### 6.4.15 CPUArchitectureName 要素

##### 6.4.15.1 定義

この要素は、実行環境にあるジョブが必要とする CPU アーキテクチャを指定するトークンである。

##### 6.4.15.2 多重度

この要素の多重度は 1 である。

##### 6.4.15.3 タイプ

この要素のタイプは jsdl:ProcessorArchitectureEnumeration である。

##### 6.4.15.4 属性

属性は定義されていない。

##### 6.4.15.5 擬似スキーマ

##### 6.4.15.6 例

SPARC アーキテクチャマシンの場合。

## 6.4.16 IndividualCPUSpeed 要素

### 6.4.16.1 定義

この要素は、実行環境にあるジョブが必要とする各 CPU のスピードを指定するレンジ値である。この要素はヘルツを単位に与えられる。この要素が存在しない場合は定義が行われず、システムがいかなる値でも選ぶことができる。

### 6.4.16.2 多重度

この要素の多重度は 0 または 1 である。

### 6.4.16.3 タイプ

この要素のタイプは `jsdl:RangeValue_Type` である。

### 6.4.16.4 属性

属性は定義されていない。

### 6.4.16.5 擬似スキーマ

### 6.4.16.6 例

1 ギガヘルツ以上のスピードを持つ CPU の場合。

## 6.4.17 IndividualCPUTime 要素

### 6.4.17.1 定義

この要素は、ジョブの実行のために各リソースで求められる CPU 時間の合計を指定するレンジ値である。この要素が存在しない場合は定義が行われず、システムがいかなる値でも選ぶことができる。

### 6.4.17.2 多重度

この要素の多重度は 0 または 1 である。

### 6.4.17.3 タイプ

この要素のタイプは `jsdl:RangeValue_Type` である。

### 6.4.17.4 属性

属性は定義されていない。

### 6.4.17.5 擬似スキーマ

### 6.4.17.6 例

最大 60 秒の CPU 時間の場合。

## 6.4.18 IndividualCPUCount 要素

### 6.4.18.1 定義

この要素は、ジョブサブミッションにアロケートする各リソースに対し、与える CPU の数を指定するレンジ値である。この要素が存在しない場合は定義が行われず、システムがいかなる値でも選ぶことができる。

### 6.4.18.2 多重度

この要素の多重度は 0 または 1 である。

#### 6.4.18.3 タイプ

この要素のタイプは jsdl:RangeValue\_Type である。

#### 6.4.18.4 属性

属性は定義されていない。

#### 6.4.18.5 擬似スキーマ

#### 6.4.18.6 例

与えられた各リソースがこのジョブに対して 2 つの CPU を用意する場合。

### 6.4.19 IndividualNetworkBandwidth 要素

#### 6.4.19.1 定義

この要素は、個々のリソースの帯域幅に関する要件を指定するレンジ値である。その値はビット毎秒を単位に与えられる。この要素が存在しない場合は定義が行われず、システムがいかなる値でも選ぶことができる。

#### 6.4.19.2 多重度

この要素の多重度は 0 または 1 である。

#### 6.4.19.3 タイプ

この要素のタイプは jsdl:RangeValue\_Type である。

#### 6.4.19.4 属性

属性は定義されていない。

#### 6.4.19.5 擬似スキーマ

#### 6.4.19.6 例

ネットワーク帯域幅が 100 メガビット毎秒以上のリソースの場合。

### 6.4.20 IndividualPhysicalMemory 要素

#### 6.4.20.1 定義

この要素は、個々のリソースで必要とする物理メモリの大きさを指定するレンジ値である。その値はバイトを単位に与えられる。この要素が存在しない場合は定義が行われず、システムがいかなる値でも選ぶことができる。

#### 6.4.20.2 多重度

この要素の多重度は 0 または 1 である。

#### 6.4.20.3 タイプ

この要素のタイプは jsdl:RangeValue\_Type である。

#### 6.4.20.4 属性

属性は定義されていない。

#### 6.4.20.5 擬似スキーマ

#### 6.4.20.6 例

各リソースが 1 ギガバイト以上の物理メモリを持つ必要がある場合。

#### 6.4.21 IndividualVirtualMemory 要素

##### 6.4.21.1 定義

この要素は、ジョブサブミッションにアロケートする各リソースで必要とする仮想メモリの大きさを指定するレンジ値である。その値はバイトを単位に与えられる。この要素が存在しない場合は定義が行われず、システムがいかなる値でも選ぶことができる。

##### 6.4.21.2 多重度

この要素の多重度は 0 または 1 である。

##### 6.4.21.3 タイプ

この要素のタイプは `jsdl:RangeValue_Type` である。

##### 6.4.21.4 属性

属性は定義されていない。

##### 6.4.21.5 擬似スキーマ

##### 6.4.21.6 例

あるリソースが 1 ギガバイト以上の仮想メモリを持っている場合。

#### 6.4.22 IndividualDiskSpace 要素

##### 6.4.22.1 定義

この要素は、ジョブサブミッションにアロケートする各リソースで必要とするディスクスペースの大きさを指定するレンジ値である。その値はバイトを単位に与えられる。この要素が存在しない場合は定義が行われず、システムがいかなる値でも選ぶことができる。

##### 6.4.22.2 多重度

この要素の多重度は 0 または 1 である。

##### 6.4.22.3 タイプ

この要素のタイプは `jsdl:RangeValue_Type` である。

##### 6.4.22.4 属性

属性は定義されていない。

##### 6.4.22.5 擬似スキーマ

##### 6.4.22.6 例

各リソースに 1 ギガバイト以上のディスクスペースが必要な場合。  
この例では、ジョブにリソースを用意するため、別のコンフィグレーションメカニズムが必要になるかもしれない点に注意。

#### 6.4.23 TotalCPUTime 要素

#### 6.4.23.1 定義

この要素は、ジョブの実行に使用するすべての CPU に関し、必要な全 CPU 時間（秒）を指定するレンジ値である。この要素が存在しない場合は定義が行われず、システムがいかなる値でも選ぶことができる。

#### 6.4.23.2 多重度

この要素の多重度は 0 または 1 である。

#### 6.4.23.3 タイプ

この要素のタイプは `jsdl:RangeValue_Type` である。

#### 6.4.23.4 属性

属性は定義されていない。

#### 6.4.23.5 擬似スキーマ

#### 6.4.23.6 例

すべての CPU に関し、CPU 時間が合計 600 秒以下になる場合。

### 6.4.24 TotalCPUCount 要素

#### 6.4.24.1 定義

この要素は、ジョブサブミッションに必要な全 CPU 数を指定するレンジ値である。この要素が存在しない場合は定義が行われず、システムがいかなる値でも選ぶことができる。

#### 6.4.24.2 多重度

この要素の多重度は 0 または 1 である。

#### 6.4.24.3 タイプ

この要素のタイプは `jsdl:RangeValue_Type` である。

#### 6.4.24.4 属性

属性は定義されていない。

#### 6.4.24.5 擬似スキーマ

#### 6.4.24.6 例

あわせて 2 つの CPU が必要な場合。

### 6.4.25 TotalPhysicalMemory 要素

#### 6.4.25.1 定義

この要素は、すべてのリソースにわたってジョブ全体で必要とする物理メモリを指定するレンジ値であり、単位はバイトで与えられる。この要素が存在しない場合は定義が行われず、システムがいかなる値でも選ぶことができる。

#### 6.4.25.2 多重度

この要素の多重度は 0 または 1 である。

#### 6.4.25.3 タイプ

この要素のタイプは jsdl:RangeValue\_Type である。

#### 6.4.25.4 属性

属性は定義されていない。

#### 6.4.25.5 擬似スキーマ

#### 6.4.25.6 例

リソース間であわせて 10 ギガバイトの物理メモリを必要とする場合。

### 6.4.26 TotalVirtualMemory 要素

#### 6.4.26.1 定義

この要素は、ジョブサブミッションに必要な仮想メモリの総量を指定するレンジ値であり、単位はバイトで与えられる。この要素が存在しない場合は定義が行われず、システムがいかなる値でも選ぶことができる。

#### 6.4.26.2 多重度

この要素の多重度は 0 または 1 である。

#### 6.4.26.3 タイプ

この要素のタイプは jsdl:RangeValue\_Type である。

#### 6.4.26.4 属性

属性は定義されていない。

#### 6.4.26.5 擬似スキーマ

#### 6.4.26.6 例

最低 1 ギガバイトの仮想メモリを持つリソースの場合。

### 6.4.27 TotalDiskSpace 要素

#### 6.4.27.1 定義

この要素は、ジョブにアロケートする必要のあるディスクスペースの総量を指定するレンジ値であり、単位はバイトで与えられる。この要素が存在しない場合は定義が行われず、システムがいかなる値でも選ぶことができる。

#### 6.4.27.2 多重度

この要素の多重度は 0 または 1 である。

#### 6.4.27.3 タイプ

この要素のタイプは jsdl:RangeValue\_Type である。

#### 6.4.27.4 属性

属性は定義されていない。

#### 6.4.27.5 擬似スキーマ

#### 6.4.27.6 例

ジョブが合計で最低 1 ギガバイトのディスクスペースを必要とする場合。

この例では、ジョブにリソースを用意するため、別のコンフィグレーションメカニズムが必要になるかもしれない点に注意。

#### 6.4.28 TotalResourceCount 要素

##### 6.4.28.1 定義

この要素は、ジョブが必要とするリソースの合計数を指定するレンジ値である。この要素が存在しない場合は定義が行われず、システムがいかなる値でも選ぶことができる。

##### 6.4.28.2 多重度

この要素の多重度は 0 または 1 である。

##### 6.4.28.3 タイプ

この要素のタイプは `jsdl:RangeValue_Type` である。

##### 6.4.28.4 属性

属性は定義されていない。

##### 6.4.28.5 擬似スキーマ

##### 6.4.28.6 例

5 つのリソースを必要とする場合。

5 つのリソースの場合。ただしジョブに対してそれぞれ 2 つの CPU を持つ。

#### 6.4.29 追加リソース

JSDL を拡張して (7 章参照) 追加リソースを記述することができる。たとえば、ライセンス、名前の付けられたリソース、ソフトウェアライブラリ、ソフトウェアパッケージ、特別なハードウェアなどは、拡張を使って記述することができる、

#### 6.5 データステージング要素

##### 6.5.1 DataStaging 要素

###### 6.5.1.1 定義

データステージングは、実行ホストに移動させる (ステージインする) 必要のあるファイルと、実行ホストから移動させる (ステージアウトする) 必要のあるファイルを定義するものである。ファイルはジョブが実行を開始する前にステージインし、ジョブが終了した後にステージアウトする。

FileName 要素や Source 要素にディレクトリが指定されている場合は、再帰コピーが行われる。実行環境が再帰コピーをサポートしていなければ、エラーが報告される。エラーがいつどのように発生したかという点も含め、このエラーの仕様は JSDL 仕様の対象外である。

複数のステージアウト用 DataStaging 要素に (同じ FileSystem 上の) 同じ FileName を指定することで、同じファイルを複数回ステージアウトすることが可能である。

別々の DataStaging 要素で同じ FileName を使って同じローカルファイルにステージインすることもできる。ただしこれは推奨できず、結果は未確定な部分が多い。

CreationFlag は、ステージされたファイルが既存のファイルに追加されるのか、あるいは既存のファイルの上書きされるのかを決めるものである。この要素は DataStaging 要素の中に存在していなければならない。

DeleteOnTermination 要素を使用すると、ジョブの終了後にファイルを削除することができる。ファイルがステージアウトされる場合は、その作業が終わってから削除が行われる。また、ファイルがジョブによって作成されたものでなくても削除される。たとえば実行ホストにあるファイルが存在し、CreationFlag が dontOverwrite に設定されているものとする。この場合でも、ジョブが適切な権限を持っていて DeleteOnTermination が真であれば、そのファイルは削除される。

JSDL 文書中の DataStaging 要素の順番は重要ではない。すなわち、文書に登場する DataStaging 要素の順序に意味はない。ただし、ジョブの実行順序や、複数のステージイン（あるいはステージアウト）作業の実行順序がすでに決められている場合は別である。

より複雑なファイル転送は、本仕様書の範囲外である。たとえば、グリッド型のファイルシステムにおけるジョブ終了時のステータスや予備段階に基づく条件付トランスファーなどがそうである。

ステージされたファイルに対する許可やアクセス制御は、実装が扱うべきもので、JSDL 仕様の範囲外である。

ここに示したステージングよりも複雑なデプロイメントシナリオ、たとえば実行環境そのもののデプロイメントやコンフィグレーションは、JSDL 仕様の範囲外である。

#### 6.5.1.2 多重度

この要素の多重度は 0 または 1 である。

#### 6.5.1.3 タイプ

これは複雑型であり、以下の要素をサポートしていなければならない。

- FileName
- FileSystemName
- CreationFlag
- DeleteOnTermination
- Source
- Target

#### 6.5.1.4 属性

以下の属性が定義されている。

• name: DataStaging 要素に対するオプション名。タイプは xsd:NCName。この名前はユーザが定義し、JSDL 文書内において固有のものでなければならない。

#### 6.5.1.5 擬似スキーマ

#### 6.5.1.6 例

ファイルのステージング

FileSystem に相対的なファイルのステージング

ステージインとステージアウト。ジョブの実行前にファイルのステージインが行われ、ジョブの終了後にファイルのステージアウトが行われるよう、Source と Target の両方の要素を定義することができる。

別々の DataStaging 要素で ( 同じ FileSystem の ) 同じ FileName を用いることで、複数のステージアウト作業を指定することができる。

#### 6.5.2 FileName 要素

##### 6.5.2.1 定義

この要素は、実行ホスト上のファイル (あるいはディレクトリ) のローカル名を指定する文字列である。FileName は相対パス (6.5.3 節参照)<sup>8</sup> で書き、区切り記号には"/"のみを使用しなければならない。すなわち FileName は"/"から始まるものであってはならない。FileName は<ディレクトリ>/<ディレクトリ>/.../<名前>という階層構造をしたディレクトリパスの形をとっていてもよい。"<名前>"は、ディレクトリかファイルのいずれかである。

##### 6.5.2.2 多重度

この要素の多重度は 1 である。

##### 6.5.2.3 タイプ

この要素のタイプは xsd:string である。

##### 6.5.2.4 属性

属性は定義されていない。

##### 6.5.2.5 擬似スキーマ

##### 6.5.2.6 例

ファイル名

ファイルを指定する階層ディレクトリパス

ディレクトリを指定する階層ディレクトリパス

#### 6.5.3 FileSystemName 要素

##### 6.5.3.1 定義

FileSystemName が与えられると、その名前前で参照される FileSystem 宣言に対して FileName が相対的なものになる。この場合、その名前を持った FileSystem が実際に存在していなければならない。FileSystemName が与えられない場合は定義が行われず、定義が行われなければ FileName は、システムが決めるワーキングジョブディレクトリに対して相対的に与えられる。なお、JSDL 拡張を使えば、ワーキングジョブディレクトリに値

を定義することができる。たとえば“Executables on POSIX Conformant Hosts”（POSIX 準拠ホスト上の実行ファイル）拡張の WorkingDirectory 要素の定義（8.1.7 節）を参照のこと。

-----  
<sup>8</sup>FileName は、背景にあるシステムのファイルシステムツリーのルートディレクトリに対応させた FileSystem に対して相対的に与えることができるため、このことは拡張性においてあまり重大な制限とはならない。

#### 6.5.3.2 多重度

この要素の多重度は 0 または 1 である。

#### 6.5.3.3 タイプ

この要素のタイプは xsd:NCName である。

#### 6.5.3.4 属性

属性は定義されていない。

#### 6.5.3.5 擬似スキーマ

#### 6.5.3.6 例

特定の FileSystem にファイルをステージングする場合。

### 6.5.4 CreationFlag 要素

#### 6.5.4.1 定義

この要素は、ローカルの実行システムに作成されるファイルを既存のファイルに上書きするか追加するかを決めるものである。多くの場合で“overwrite”（上書き）とするのが普通である。

#### 6.5.4.2 多重度

この要素の多重度は 1 である。

#### 6.5.4.3 タイプ

この要素のタイプは jsdl:CreationFlagEnumeration である。

#### 6.5.4.4 属性

属性は定義されていない。

#### 6.5.4.5 擬似スキーマ

#### 6.5.4.6 例

既存のファイルに上書きする場合。

### 6.5.5 DeleteOnTermination 要素

#### 6.5.5.1 定義

この論理型はジョブの終了後にファイルを削除するかどうかを決めるものである。これが真の場合は、ジョブの終了後あるいはファイルがステージアウトされた後で、ファイルは削除される。偽の場合は、ファイルの置かれた FileSystem が存続している限り、そのファイルは残る。この要素が存在しない場合は、ファイルの削除に関して何も指定されず、システムにまかされることになる。

#### 6.5.5.2 多重度

この要素の多重度は 0 または 1 である。

#### 6.5.5.3 タイプ

この要素のタイプは `xsd:boolean` である。

- ・ 真：ジョブ終了後にファイルを削除する
- ・ 偽：ジョブ終了後にファイルを削除しない

#### 6.5.5.4 属性

属性は定義されていない。

#### 6.5.5.5 擬似スキーマ

#### 6.5.5.6 例

ジョブの終了後にファイルを削除する場合。

### 6.5.6 Source 要素

#### 6.5.6.1 定義

Source 要素は、リモートシステム上のファイルやディレクトリの場所を指定するものである。このファイルやディレクトリは、ジョブが始まる前に（オプションの）URI が指定する場所からステージインされなければならない。この要素が存在しない場合は、ファイルをステージインする必要はない。

#### 6.5.6.2 多重度

この要素の多重度は 0 または 1 である。

#### 6.5.6.3 タイプ

この要素は複合型であり、以下の要素をサポートしていなければならない。

- ・ URI 要素：[RFC2396]にしたがった URI

#### 6.5.6.4 属性

属性は定義されていない。

#### 6.5.6.5 擬似スキーマ

#### 6.5.6.6 例

ソースが 1 つのファイルである場合。

ソースがディレクトリである場合。

URI は `http://`だけでなくプロトコルも指定できる。

## 6.5.7 URI 要素

### 6.5.7.1 定義

これは URI [RFC2396]である。この要素は、ファイルをステージインあるいはステージアウトするために使用する場所（およびプロトコル）を指定できる。

### 6.5.7.2 多重度

この要素の多重度は 0 または 1 である。

### 6.5.7.3 タイプ

この要素のタイプは xsd:anyURI である。

### 6.5.7.4 属性

属性は定義されていない。

### 6.5.7.5 擬似スキーマ

## 6.5.8 Target 要素

### 6.5.8.1 定義

Target 要素はリモートシステム上のファイルやディレクトリの場所を指定するものである。このファイルやディレクトリは、ジョブが終わった後に（オプションの）URI が指定する場所からステージアウトされなければならない。この要素が存在しない場合は、ファイルをステージアウトする必要はない。

### 6.5.8.2 多重度

この要素の多重度は 0 または 1 である。

### 6.5.8.3 タイプ

この要素は複合型であり、以下の要素をサポートしていなければならない。

- ・ URI 要素：[RFC2396]にしたがった URI

### 6.5.8.4 属性

属性は定義されていない。

### 6.5.8.5 擬似スキーマ

### 6.5.8.6 例

対象が 1 つのファイルである場合。

対象がディレクトリである場合。

## 7 JSDL の拡張

JSDL は、ジョブのサブミッション要件を定義する全体的な構造を与えるものである。この構造は特定のニーズに最大限に合うよう拡張することができる。JSDL では、拡張に関して属性を使う場合と要素を使う場合の 2 つのメカニズムがある。一般に拡張メカニズムを使うことにより相互運用性が低下するため、これらのメカニズムは必要な場合のみ慎重に使用すべきである。JSDL を拡張する際は、要件に合致する拡張を他のグループがすでに提供済みでないことを最初に確認するべきである。

拡張が定義する要素や属性が JSDL 文書に存在している場合、これらを JSDL の要素や属性と同じようにサポートする必要がある。

また拡張が定義する要素や属性が JSDL 文書に存在している場合、文書全体が満たされるためにはこれらも満足されなければならない。

ただし JSDL 文書をシステムにサブミットした結果は JSDL 仕様の対象外であることに注意が必要である。

## 7.1 属性拡張

各 JSDL 要素は、属性を必要な数だけ追加することができる。ただしこれらの属性が JSDL が定義する規範的ネームスペース以外のネームスペースを持っている場合に限る。以下の例(簡単にするため短くしている)では、“order”と呼ばれる属性を導入している。order はファイルのステージングの順番を定義するものである。

## 7.2 要素拡張

JSDL スキーマは、JSDL が規範的に定義していない要素を、文書の全体的な構造の中の適当な場所に追加することができる。属性拡張と同様に、これらの要素は、JSDL が定義する規範的ネームスペース以外のネームスペースを持っている必要がある。以下の例(簡単にするため短くしている)では、JSDL を拡張してリソース予約を導入している。

## 7.3 JSDL “other”値のセマンティクス

JSDL は、プロセッサアーキテクチャとオペレーティングシステムに対してエニュメレーションを定義している。どちらのエニュメレーションもすべてを網羅しているとは想定していない。これらのエニュメレーションはラップタイプで埋め込まれ、それぞれ“jsdl:CPUArchitecture\_Type”と“jsdl:OperatingSystemType\_Type”という拡張を使用できるようにする。

これらのエニュメレーションを拡張するため、それぞれ“jsdl:CPUArchitectureName”と“jsdl:OperatingSystemName”という要素の値に“other”という特別なキーワードを使用しなければならない。上述の要素拡張パターンも使用しなければならない。

たとえば“JavaCPU”という名の Java バイトコードをネイティブで実行する CPU があるとする。この CPU は、JSDL 文書のリソースのセクションで、以下のように指定することができる。

## 8 規範的拡張

以下の拡張は JSDL 1.0 が規範的に定義したものである。

### 8.1 POSIX 準拠ホスト上の実行ファイル

この規範的拡張は、POSIX 準拠システム上で実行するアプリケーションを記述するスキ

ーマを定義するものである。

本仕様書では、“jsdl-posix”をこのスキーマに使用するネームスペースプリフィクスとする。このスキーマの規範的ネームスペースは表 2 - 1 に与えられている。

#### 8.1.1 POSIXApplication 要素

##### 8.1.1.1 定義

この要素は POSIX スタイルのアプリケーションとその要件を記述するものである。Executable、Argument、Input、Output、Error、WorkingDirectory、Environment の各要素に加え、さまざまな POSIX リミット要素や UserName、GroupName などの要素も含んでいる。この要素が JSDL の Application 要素の下位要素として存在している場合は、一度しか使用できない。

##### 8.1.1.2 多重度

この要素の多重度は 0 または 1 である。

##### 8.1.1.3 タイプ

この要素は複合型であり、以下の要素をサポートしていなければならない。

##### 8.1.1.4 属性

次の属性が定義されている。

- ・ name: POSIXApplication 要素に対するオプション名。タイプは xsd:NCName であるため、これを含んだ JSDL 文書の外部から再使用したり参照したりすることができる。

##### 8.1.1.5 擬似スキーマ

#### 8.1.2 Executable 要素

##### 8.1.2.1 定義

この要素は、実行するコマンドを指定する文字列である。ApplicationName 要素と Executable 要素がともに与えられていない場合、JSDL 文書はヌルジョブあるいはデータステージングを定義する。どちらも与えられている場合は、Executable は ApplicationVersion 要素（存在する場合）の内容に応じて、名前をつけられたアプリケーションの実行に使用する実行ファイルを指定しなければならない。6.3.1 節および 6.3.2 節も参照のこと。

##### 8.1.2.2 多重度

この要素の多重度は 0 または 1 である。

##### 8.1.2.3 タイプ

この要素のタイプは xsd:string である。

##### 8.1.2.4 属性

次の属性が定義されている。

- ・ filesystemName: JSDL 文書内の FileSystem 要素で定義されたファイルシステムの名前。これが存在している場合、Executable 要素の文字列内容を、名前をつけられた FileSystem

のマウントポイントに相対的なファイル名として解釈しなければならない。

#### 8.1.2.5 擬似スキーマ

#### 8.1.2.6 例

あらわに名前をつけられた実行ファイルの場合。

コンシューマが定義した方法で扱う場合。

FileSystem “HOME” に対して相対的な場所にある実行ファイル。通常、“~/scripts/myExample”という形に書かれる。

### 8.1.3 Argument 要素

#### 8.1.3.1 定義

この要素は制限付きの規格化された文字列であり、アプリケーションに引数要素を指定するものである。Argument 要素は空であっても構わないが、コラプスさせてはならない。

#### 8.1.3.2 多重度

この要素の多重度は 0 または 1 である。

#### 8.1.3.3 タイプ

この要素のタイプは xsd:normalizedString である。

#### 8.1.3.4 属性

次の属性が定義されている。

-----  
⁹たとえば PATH 上で実行ファイルを探す。

・filesystemName: JSDL 文書内の FileSystem 要素で定義されたファイルシステムの名前。これが存在している場合、Argument 要素の文字列内容を、名前をつけられた FileSystem のマウントポイントに相対的なファイル名あるいはディレクトリ名として解釈しなければならない。Argument 要素の本体が空で、さらに filesystemName を持っている場合、アプリケーションへの引数は名前をつけられた FileSystem のマウントポイントでなければならない。

#### 8.1.3.5 擬似スキーマ

#### 8.1.3.6 例

Java アプリケーションに“CLASSPATH”と起動するクラスを指定する場合。

この例は'[java] -cp ./example.jar.org.example.Main'と解釈される。

Administrator ではないユーザが Windows 上で Apache2 サービスを開始する場合。

この例は'[runas] /user:Administrator@WORKGROUUP"net start Apache2"'と解釈される。

引数の連結リストを stdout に echo する場合。

この例は'[echo]foo bar baz'と解釈される。

この例は'[echo]foo bar "" baz'と解釈される。

この例は'[echo]foo bar baz "'と解釈される。

ユーザのホームディレクトリの場所を stdout に echo する場合。

この例は'[echo] /home/darrenp'と解釈される(ここで HOME が/home/darrenp にマウントされているものとする)。

## 8.1.4 Input 要素

### 8.1.4.1 定義

この要素は、コマンドへの入力(標準入力)を表す文字列である。Input 要素は、ワーキングディレクトリまたは名前を付けられた FileSystem に相対的なファイル名である。この要素が存在しない場合は定義が行われず、システムがいかなる値でも選ぶことができる。

### 8.1.4.2 多重度

この要素の多重度は 0 または 1 である。

### 8.1.4.3 タイプ

この要素のタイプは xsd:string である。

### 8.1.4.4 属性

次の属性が定義されている。

・filesystemName: JSDL 文書内の FileSystem 要素に定義されたファイルシステムの名前。これが存在する場合は、Input 要素の内容に書かれた文字列を、名前の付けられたファイルシステムのマウントポイントに相対的なファイル名として解釈しなければならない。

### 8.1.4.5 擬似スキーマ

### 8.1.4.6 例

標準入力を "~/input.txt" から取り出す場合。

## 8.1.5 Output 要素

### 8.1.5.1 定義

この要素は、コマンドへの出力(標準出力)を表す文字列である。Output 要素は、ワーキングディレクトリまたは名前を付けられた FileSystem に相対的なファイル名である。この要素が存在しない場合は定義が行われず、システムがいかなる値でも選ぶことができる。

### 8.1.5.2 多重度

この要素の多重度は 0 または 1 である。

### 8.1.5.3 タイプ

この要素のタイプは xsd:string である。

### 8.1.5.4 属性

次の属性が定義されている。

・filesystemName: JSDL 文書内の FileSystem 要素に定義されたファイルシステムの名前。これが存在する場合は、Output 要素の内容に書かれた文字列を、名前の付けられたファイルシステムのマウントポイントに相対的なファイル名として解釈しなければならない。

#### 8.1.5.5 擬似スキーマ

#### 8.1.5.6 例

標準出力を”~/output.txt”に書き込む場合。

### 8.1.6 Error 要素

#### 8.1.6.1 定義

この要素は、コマンドへのエラー出力(標準エラー)を表す文字列である。Error 要素は、ワーキングディレクトリまたは名前を付けられた FileSystem に相対的なファイル名である。この要素が存在しない場合は定義が行われず、システムがいかなる値でも選ぶことができる。

#### 8.1.6.2 多重度

この要素の多重度は 0 または 1 である。

#### 8.1.6.3 タイプ

この要素のタイプは xsd:string である。

#### 8.1.6.4 属性

次の属性が定義されている。

- filesystemName: JSDDL 文書内の FileSystem 要素に定義されたファイルシステムの名前。これが存在する場合は、Error 要素の内容に書かれた文字列を、名前の付けられたファイルシステムのマウントポイントに相対的なファイル名として解釈しなければならない。

#### 8.1.6.5 擬似スキーマ

#### 8.1.6.6 例

標準エラーを”~/error.txt”に書き込む場合。

### 8.1.7 WorkingDirectory 要素

#### 8.1.7.1 定義

この要素は、ジョブの実行に必要な開始ディレクトリ(ワーキングディレクトリ)を表す文字列である。この要素が存在しない場合は定義が行われず、システムがいかなる値でも選ぶことができる。多くの場合ワーキングディレクトリは、WorkingDirectory 要素を FileSystem 要素のローカルマウントパスと等しくなるように指定することで、FileSystem 要素と関連づけることができる。WorkingDirectory 要素は、特定の FileSystem 要素に対し、必要とされる直接の関連性を持たないこともある。

#### 8.1.7.2 多重度

この要素の多重度は 0 または 1 である。

#### 8.1.7.3 タイプ

この要素のタイプは xsd:string である。

#### 8.1.7.4 属性

次の属性が定義されている。

・filesystemName: JSDL 文書内の FileSystem 要素に定義されたファイルシステムの名前。これが存在する場合は、WorkingDirectory 要素の内容に書かれた文字列を、名前の付けられたファイルシステムのマウントポイントに相対的なディレクトリ名として解釈しなければならない。

#### 8.1.7.5 擬似スキーマ

#### 8.1.7.6 例

ジョブのワーキングディレクトリがユーザのホームディレクトリである場合。

### 8.1.8 Environment 要素

#### 8.1.8.1 定義

この要素は、実行環境でジョブに定義される環境変数の名前と値を指定するものである。

特別な場合として、環境変数は JSDL 文書内で宣言されたファイルシステムのマウントポイントを含むように宣言することができる。これは Environment 要素の filesystemName 属性と FileSystem 要素の名前を使って行う。以下に示す例を参照のこと。また 6.4.22 節を参照のこと。

#### 8.1.8.2 多重度

この要素の多重度は 0 または 1 である。

#### 8.1.8.3 タイプ

この要素のタイプは xsd:string である。これは環境変数の値である。

#### 8.1.8.4 属性

次の属性が定義されている。

・ name: 環境変数の名前

・filesystemName: JSDL 文書内の FileSystem 要素に定義されたファイルシステムの名前。これが存在する場合は、Environment 要素の内容に書かれた文字列を、名前の付けられたファイルシステムのマウントポイントに相対的なファイル名またはディレクトリ名として解釈しなければならない。

#### 8.1.8.5 擬似スキーマ

#### 8.1.8.6 例

SHELL を bash に設定する場合。

MAIL の場所を "~/mailboxes/INBOX" に設定する場合。

環境変数 "TMP DIR" を "TMP" という名のファイルシステムのルートに設定する場合。

### 8.1.9 WallTimeLimit 要素

#### 8.1.9.1 定義

この要素は、アプリケーションの実行時間をソフトウェア的に制限するものであり、秒

数を正の整数で表す。この要素がない場合はシステムはデフォルト値を使用できる。

#### 8.1.9.2 多重度

この要素の多重度は 0 または 1 である。

#### 8.1.9.3 タイプ

この要素のタイプは `xsd:nonNegativeInteger` である。

#### 8.1.9.4 属性

属性は定義されていない。

#### 8.1.9.5 擬似スキーマ

#### 8.1.9.6 例

実行時間を 1 分に設定する場合。

### 8.1.10 FileSizeLimit 要素

#### 8.1.10.1 定義

この要素は、ジョブに関連したファイルの最大サイズを与える正の整数値であり、ファイルのサイズはバイトで与えられる。この要素が存在しない場合はシステムはデフォルト値を使用できる。

#### 8.1.10.2 多重度

この要素の多重度は 0 または 1 である。

#### 8.1.10.3 タイプ

この要素のタイプは `xsd:nonNegativeInteger` である。

#### 8.1.10.4 属性

属性は定義されていない。

#### 8.1.10.5 擬似スキーマ

#### 8.1.10.6 例

最大ファイルサイズを 1 ギガバイトに設定する場合。

### 8.1.11 CoreDumpLimit 要素

#### 8.1.11.1 定義

この要素は、ジョブが生成する可能性のあるコアダンプの最大サイズを表す正の整数値であり、サイズはバイトで与えられる。この要素が存在しない場合はシステムはデフォルト値を使用できる。

#### 8.1.11.2 多重度

この要素の多重度は 0 または 1 である。

#### 8.1.11.3 タイプ

この要素のタイプは `xsd:nonNegativeInteger` である。

#### 8.1.11.4 属性

属性は定義されていない。

#### 8.1.11.5 擬似スキーマ

#### 8.1.11.6 例

サイズを 0 に設定して、コアダンプを無効とする場合。

#### 8.1.12 DataSegmentLimit 要素

##### 8.1.12.1 定義

この要素は、データセグメントを与えられたサイズに制限するものであり、正の整数で表す。単位はバイトで与えられる。この要素が存在しない場合はシステムはデフォルト値を使用できる。

##### 8.1.12.2 多重度

この要素の多重度は 0 または 1 である。

##### 8.1.12.3 タイプ

この要素のタイプは `xsd:nonNegativeInteger` である。

##### 8.1.12.4 属性

属性は定義されていない。

#### 8.1.12.5 擬似スキーマ

#### 8.1.12.6 例

データセグメントを 32 キロバイトに制限する場合。

#### 8.1.13 LockedMemoryLimit 要素

##### 8.1.13.1 定義

この要素は、ジョブがロックすることを許されている物理メモリの最大量を正の整数で表すものである。単位はバイトで与えられる。この要素が存在しない場合はシステムはデフォルト値を使用できる。

##### 8.1.13.2 多重度

この要素の多重度は 0 または 1 である。

##### 8.1.13.3 タイプ

この要素のタイプは `xsd:nonNegativeInteger` である。

##### 8.1.13.4 属性

属性は定義されていない。

#### 8.1.13.5 擬似スキーマ

#### 8.1.13.6 例

最大 8 メガバイトまでメモリをロックできるとする場合。

#### 8.1.14 MemoryLimit 要素

#### 8.1.14.1 定義

この要素は、ジョブが実行時に使用する物理メモリの最大量を正の整数で表すものである。単位はバイトで与えられる。この要素が存在しない場合はシステムはデフォルト値を使用できる<sup>10</sup>。

#### 8.1.14.2 多重度

この要素の多重度は0または1である。

#### 8.1.14.3 タイプ

この要素のタイプは `xsd:nonNegativeInteger` である。

#### 8.1.14.4 属性

属性は定義されていない。

#### 8.1.14.5 擬似スキーマ

#### 8.1.14.6 例

物理メモリを 64 メガバイトに設定する場合。

### 8.1.15 OpenDescriptorsLimit 要素

#### 8.1.15.1 定義

この要素は、ジョブが持つことのできるオープンファイル記述子（ファイル、パイプ、ソケット）の最大数を正の整数で表すものである。この要素が存在しない場合はシステムはデフォルト値を使用できる。

#### 8.1.15.2 多重度

この要素の多重度は0または1である。

#### 8.1.15.3 タイプ

この要素のタイプは `xsd:nonNegativeInteger` である。

#### 8.1.15.4 属性

属性は定義されていない。

-----  
<sup>10</sup> これはレジデントセットのサイズ制限であり、背景にあるシステムがキロバイトの粒度を課すことがある。

#### 8.1.15.5 擬似スキーマ

#### 8.1.15.6 例

記述子の最大数を 16 とする場合。

### 8.1.16 PipeSizeLimit 要素

#### 8.1.16.1 定義

この要素は、ジョブの処理が生成するパイプラインの最大サイズを正の整数で記述するものである。単位はバイトで与えられる。この要素が存在しない場合はシステムはデフォ

ルト値を使用できる<sup>11</sup>。

#### 8.1.16.2 多重度

この要素の多重度は0または1である。

#### 8.1.16.3 タイプ

この要素のタイプは `xsd:nonNegativeInteger` である。

#### 8.1.16.4 属性

属性は定義されていない。

#### 8.1.16.5 擬似スキーマ

#### 8.1.16.6 例

512 バイトの制限をつける場合。

### 8.1.17 StackSizeLimit 要素

#### 8.1.17.1 定義

この要素は、ジョブの実行スタックの最大サイズを正の整数で表したものである。単位はバイトで与えられる。この要素が存在しない場合はシステムはデフォルト値を使用できる。

#### 8.1.17.2 多重度

この要素の多重度は0または1である。

#### 8.1.17.3 タイプ

この要素のタイプは `xsd:nonNegativeInteger` である。

#### 8.1.17.4 属性

属性は定義されていない。

#### 8.1.17.5 擬似スキーマ

-----  
<sup>11</sup> どんなシステムでもパイプバッファを任意の大きさとすることはできないが、デフォルトをもうけることは通常何の問題もない。多くのシステムは512バイトの粒度を課す。

#### 8.1.17.6 例

スタックサイズを1メガバイトとする場合。

### 8.1.18 CPUTimeLimit 要素

#### 8.1.18.1 定義

この要素は、SIGXCPU 信号がジョブに送られる前にジョブが消費を許された CPU 時間を正の整数で表したものである。単位は秒で与えられる。この要素が存在しない場合は定義が行われず、システムはデフォルト値を使用できる。

#### 8.1.18.2 多重度

この要素の多重度は0または1である。

#### 8.1.18.3 タイプ

この要素のタイプは `xsd:nonNegativeInteger` である。

#### 8.1.18.4 属性

属性は定義されていない。

#### 8.1.18.5 擬似スキーマ

#### 8.1.18.6 例

30 秒の CPU 時間を許可する場合。

### 8.1.19 ProcessCountLimit 要素

#### 8.1.19.1 定義

この要素は、ジョブが発生させることを許されたプロセスの最大数を正の整数で表したものである。この要素が存在しない場合はシステムはデフォルト値を使用できる。

#### 8.1.19.2 多重度

この要素の多重度は 0 または 1 である。

#### 8.1.19.3 タイプ

この要素のタイプは `xsd:nonNegativeInteger` である。

#### 8.1.19.4 属性

属性は定義されていない。

#### 8.1.19.5 擬似スキーマ

#### 8.1.19.6 例

最大 8 つのプロセスまで許可する場合。

### 8.1.20 VirtualMemoryLimit 要素

#### 8.1.20.1 定義

この要素は、ジョブがアロケートを許された仮想メモリの最大量を正の整数で表したものである。単位はバイトで与えられる。この要素が存在しない場合はシステムはデフォルト値を使用できる。

#### 8.1.20.2 多重度

この要素の多重度は 0 または 1 である。

#### 8.1.20.3 タイプ

この要素のタイプは `xsd:nonNegativeInteger` である。

#### 8.1.20.4 属性

属性は定義されていない。

#### 8.1.20.5 擬似スキーマ

#### 8.1.20.6 例

仮想メモリの最大量を 128 メガバイトに制限する場合。

## 8.1.21 ThreadCountLimit 要素

### 8.1.21.1 定義

この要素は、ジョブが生成を許されたスレッドの数を正の整数で表したものである。この要素が存在しない場合はシステムはデフォルト値を使用できる。

### 8.1.21.2 多重度

この要素の多重度は0または1である。

### 8.1.21.3 タイプ

この要素のタイプは `xsd:nonNegativeInteger` である。

### 8.1.21.4 属性

属性は定義されていない。

### 8.1.21.5 擬似スキーマ

### 8.1.21.6 例

スレッド数を8までとする場合。

## 8.1.22 UserName 要素

### 8.1.22.1 定義

この要素は、アプリケーションの実行時に使用するユーザ名を定義する文字列である。この要素が存在しない場合はユーザ名は定義されず、システムは実装に基づいてユーザ名を選ぶことができる。

### 8.1.22.2 多重度

この要素の多重度は0または1である。

### 8.1.22.3 タイプ

この要素のタイプは `xsd:string` である。

### 8.1.22.4 属性

属性は定義されていない。

### 8.1.22.5 擬似スキーマ

### 8.1.22.6 例

UNIX ユーザ名を使用する場合。

## 8.1.23 GroupName 要素

### 8.1.23.1 定義

この要素は、アプリケーションの実行時に使用するグループ名を定義する文字列である。この要素が存在しない場合はグループ名は定義されず、システムは実装に基づいてグループ名を選ぶことができる。

#### 8.1.23.2 多重度

この要素の多重度は 0 または 1 である。

#### 8.1.23.3 タイプ

この要素のタイプは `xsd:string` である。

#### 8.1.23.4 属性

属性は定義されていない。

#### 8.1.23.5 擬似スキーマ

#### 8.1.23.6 例

UNIX グループ名を使用する場合。

### 9 セキュリティに関して

本仕様書は、グリッドやその他のジョブ管理システムへの計算ジョブのサブミットについて、その要件を記述する言語を定義するものである。ジョブサブミッションが安全なものであることが前提となっているが、そのメカニズムは本仕様書の対象外である。

ジョブの実行に必要な権利を記述できるかどうかは、非常に重要な点である。必要な際に権利の詳細なデリゲーションを表現するため、より専門的なセキュリティ言語やポリシー言語とともに JSDL 1.0 文書を書く必要があると思われる。

#### 著者情報

##### 寄稿者

Karl Czajkowski、William Lee、Chris Smith、Kazushige Saga、David Snelling、Igor Sedukhin が本仕様書に寄稿してくれたことに深く感謝の意を表する。

##### 謝辞

本文書で扱ったトピックスに関し、議論を行った数多くの同僚に対し感謝する。とくに John Ainsworth、Sayaka Akioka、Alain Andrieux、Sven van de Berghe、Lee Cook、Asit Dan、Yuri Demchenko、Nathalie Furmento、Andreas Haas、Tomasz Haupt、Bill Harris、Hiro Kishimoto、Tom Maguire、Bill Nitzberg、Ariel Oleksiak、Jim Pruyne、Hrabri Rajic、Jennifer Schopf、Frank Siebenlist、Dan Templeton、Andrea Westerinen (アルファベット順。記載漏れがあった場合はおわびします。)

初期の草稿を呼んでコメントをくれた方々にも感謝する。貴重なコメントのおかげで本文書の読みやすさ、正確性を改善することができた。

またテレビ会議の技術的サポートに関し、Cadence Design Systems および Xango に感謝の意を表する。

#### 著作権情報

Copyright © Global Grid Forum (2003-2005). All Rights Reserved.

本文書およびその翻訳物は、複製し、他者に提供することができる。本文書に関してコメントや別の説明を与えている派生著作物、あるいは本文書の普及を助ける派生著作物についても、そのままあるいは部分的に、制約なしに作成、複製、発行、頒布が可能である。ただしそのような複製物や派生著作物については、上記の著作権情報と本段落に書かれていることを記載しなければならない。ただし、GGFや他の組織に対する著作権情報や参考文献を削除するなどといった本文書自体の変更は、いかなる形であっても禁止する。なお、グリッド提言 (Grid Recommendations) を開発する目的で変更が必要ならば、その限りではない。この場合、GGFドキュメントプロセスで決められた著作権に対する手続きを遵守する必要がある。また、英語以外の言語に翻訳する際に変更が必要な場合も、その限りではない。

上に与えた制限付き許諾は永続的なものであり、GGFあるいはその後続組織または委嘱組織が無効にすることはしない。

本文書とそこに含まれる情報は保証されたものではない。グローバル・グリッド・フォーラムは、ここにおける情報の使用がいかなる権利をも侵害していないこと、市販性、特定目的との適合性に関するいかなる黙示保証をも侵害していないことを含む (ただし必ずしもこれらに限定されない) 明示あるいは暗示の保証を拒否するものである。

#### 知的財産権について

本文書に記載された技術の実装や使用に関連すると考えられるいかなる知的財産権およびその他の権利についても、GGFは、その有効性や範囲に関して特定の立場をとるものではない。また、こうした権利下にあるいかなるライセンスについても、それが使用できるあるいは使用できない範囲について、特定の立場をとるものではない。さらに、これらのいかなる権利についても特定する努力をGGFが行うものではない。出版やライセンス保証のために用意した権利請求書、および、開発者や本仕様書の使用者が所有権を使用する上での一般的なライセンスや許可を取得する作業の結果については、GGF事務局から入手可能である。

関係者の方々は、本文書で薦められていることを実践する上で必要なあらゆる著作権、特許、特許利用、その他の所有権に対して注意を向けるようGGFは希望する。情報はGGF委員長までお寄せいただければ幸いである (GGFのウェブサイトの連絡先を参照のこと)。

#### 規範的文献

(参考文献名省略)

#### 参考となる文献

(参考文献名省略)

## 付録 1 JSDL 規範的スキーマ

本節では JSDL 1.0 に対する規範的 XML スキーマの完全な定義を示す。

このスキーマは、<https://forge.gridforum.org/projects/jsdl-wg/document/jsdl.xsd/en/17>からも読むことができる。ただし本節で与える定義と、本仕様書の別の節やウェブ上で記載されていることに違いがある場合は、本節に書かれた定義が規範となるべきものである。

## 付録 2 JSDL POSIX アプリケーション規範的スキーマ

本節では JSDL POSIX アプリケーション 1.0 に対する規範的 XML スキーマの完全な定義を示す。

このスキーマは、

<https://forge.gridforum.org/projects/jsdl-wg/document/jsdl-posix.xsd/en/5>からも読むことができる。ただし本節で与える定義と、本仕様書の別の節やウェブ上で記載されていることに違いがある場合は、本節に書かれた定義が規範となるべきものである。

## 付録 3 拡張した JSDL 参考例

以下の例は”gnuplot”アプリケーションの起動を定義するものである。この例では、JSDL 1.0 の主な要素である JobIdentification、Application、Resources、DataStaging がすべて使用されている。また POSIXApplication 拡張も使われている。これは入力ファイルと出力ファイルの両方をステージングする機能を持っている。