

gbsp エージェント・リファレンス・マニュアル

森 洋久

joshua at globalbase.org

2010-12-01 版

目次

第 1 章	はじめに	2
1.1	目的と概要	2
1.2	このマニュアルを読むために必要な知識	2
1.3	前提となるシステム要件	2
1.4	gbsp におけるインデックスの構造について	2
1.5	インデックスへのアクセス	4
1.6	登録動作概要	4
1.7	登録動作における各エージェントの役割	5
1.8	gburl の動き	6
1.9	gboar の動き	7
1.10	gbaddr の動き	7
第 2 章	基本オペレーション	8
2.1	概要	8
2.2	継承情報	8
2.3	関数	9
2.3.1	SPdatabasePath	9
2.3.2	SPdefineAgent	10
2.3.3	SPcd	11
2.3.4	SPcreatePath	12
2.3.5	SPdeletePath	13
2.3.6	SPls	14
2.3.7	SPpermitAgent	15
2.3.8	SPforbitAgent	16

第1章 はじめに

1.1 目的と概要

gbsp は GLOBALBASE LANDSCAPE サーバおよびその周辺のネットワークコンテンツを収集し、インデックス化するスパイダリング機能を有するエージェントです。gbsp は各 GLOBALBASE LANDSCAPE サーバに組み込まれ、自身のサーバの周辺を小規模にスパイダリングします。これらのデータをサーバ型 P2P 技術により、共有し、全体の情報が検索可能となります。この基本機能をこのマニュアルにまとめます。

1.2 このマニュアルを読むために必要な知識

GLOBALBASE LANDSCAPE SERVER [1] に関する知識、xl スクリプト [2] に関する知識が必要です。

1.3 前提となるシステム要件

GLOBALBASE LANDSCAPE SERVER [1]

1.4 gbsp におけるインデックスの構造について

gbsp は基本機能として、文字列を与えると、それに対応するバイナリデータを返すデータベースを提供します。これが gbsp における一番基本的なインデックスになります。文字列は辞書順による範囲検索が可能です。この範囲検索を利用し、ディレクトリ構造を実現することが出来ます。たとえば、ディレクトリ root.addr の下のデータをリストアップする場合、root.addr [st] から root.addr [end] の間を検索します。ただし、[st] はどの文字よりも小さいコードを示し、[end] は殿文字よりも大きいコードを示します。

gbsp において、内部もツリー構造をもったデータ構造によりディレクトリを直接実装しないのは、検索対象を表現するディレクトリのデリミタが各種存在し、特定できないからです。たとえば、ドメイン名の場合は「.」、ファイルシステムの場合は「/」、URL のポート番号は「:」といった具合です。

また、検索されるデータの識別子として、ディレクトリパス名 = 検索文字列そのものを利用出来ることから、ID の管理が一元化できるというメリットもあります。ディレクトリのツリー構造の途中で新たなディレクトリを作成するのが容易になります。たとえば、root.addr.ipv4 というディレクトリはあるが、root.addr というディレクトリはないという状態であった場合に、新たに、root.addr というディレクトリを構成し、その直下のディレクトリに root.addr.ipv4 を加える場合、単純に root.addr ばいう文字列をインデックスに加えるのみで対応できます。

このようなディレクトリの実装方法を導入しますと、gbsp におけるディレクトリとは、以下に述べる決まったルールに従ってインデックスに登録された文字列のセットと考えることが出来ます。

では以下に、インデックスの構造の実際を見ていきましょう。

まず、[st], [end] に相当するコード、その他のコードとして、ソースコード src/h/spidering.h 内に以下のものが定義されています。

- LC_SECC_HEADER これより小さいコードは使ってはいけません。(負のコード)

- LC_SECC_TERMINATOR これより大きいコードは使ってはいけない。(正のコード)

そのほかに、ディレクトリ検索を行うための各種コードが定義されている。

- LC_SECC_INFO_START 文字列検索結果のデータ各種を納める位置の最初を示すコード
- LC_SECC_INFO_END 文字列検索結果のデータ各種を納める位置の最後を示すコード
- LC_SECC_EMPTY_FROM 文字列の存在していない位置の最初を表すコード
- LC_SECC_EMPTY_TO 文字列の存在していない位置の最後を示すコード

文字列1の示すディレクトリおよびディレクトリ以下の情報を表す文字列の範囲は、「文字列1 LC_SECC_HEADER LC_SECC_INFO_START」から「文字列1 LC_SECC_TERMINATOR LC_SECC_INFO_END」の範囲となる。

一方、LC_SECC_EMPTY_FROM, LC_SECC_EMPTY_TO はキャッシュ時に、情報の無い範囲を示すために使われ、「文字列1 LC_SECC_TERMINATOR LC_SECC_EMPTY_FROM」から、「文字列1より大きい文字列 LC_SECC_HEADER LC_SECC_EMPTY_TO」の間にはデータがないことを示している。

最後に、ディレクトリの個別情報として、以下のコードが定義されている。

- LC_SECC_READDIR このディレクトリ配下のディレクトリ情報
- LC_SECC_AGENT このディレクトリを管理するエージェント名を指定する。

LC_SECC_READDIR は、「文字列1 ディレクトリ文字列」という文字列が存在する場合、「文字列1 LC_SECC_HEADER LC_SECC_READDIR ディレクトリ文字列」という形で登録し、「文字列1 ディレクトリ文字列」という文字列が存在すること示している。文字列1の直下のディレクトリのみを検索したい場合、「文字列1 LC_SECC_HEADER LC_SECC_READDIR LC_SECC_HEADER」から「LC_SECC_HEADER LC_SECC_READDIR LC_SECC_TERMINATOR」の間を検索すればよい。

LC_SECC_AGENT は、文字列1より下のディレクトリに管理エージェントを示している。「文字列1 LC_SECC_HEADER LC_SECC_AGENT エージェント名」という文字列として表される。

gbspcのインデックスにおいてはプリセットされている文字列(ディレクトリ)がある。以下のものである。

- root
- root.addr
- root.addr.ipv4
- root.addr.ipv6
- root.queue
- root.queue.myurl
- root.queue.url
- root.data

1.5 インデックスへのアクセス

ディレクトリに対するオペレーションは

1. インデックスを開く
2.3.1 節
2. アクティブなエージェントを gbsp に登録する。
2.3.2 節
3. カレントディレクトリを変更する。
2.3.3 節
4. ディレクトリを作成する
2.3.4 節
5. ディレクトリを削除する
2.3.5 節
6. ディレクトリに情報を書き込む
7. ディレクトリの直下のディレクトリのリストを取得する。
2.3.6 節
8. ディレクトリを管理するエージェントを加える。
2.3.7 節
9. ディレクトリを管理するエージェントを削除する。
2.3.8 節

以上である。1.~3. 以外は対象のディレクトリの管理エージェントのみが直接実行可能である。その他のエージェントがこれらのオペレーションを実行した場合、管理エージェントにその旨の情報が知らされる。管理エージェントはオペレーションをチェックし、「拒絶」、「承認後、管理エージェントによる代行実行」のいずれかが取られる。管理エージェントの動作は管理エージェントの実装依存である。

管理エージェントは一つのディレクトリに複数あってもよい。また、管理エージェントによる代行実行時に新しい管理エージェントを登録することも可能である。

1.6 登録動作概要

LANDSCAPE サーバ [1] において、gbsp を中心とした登録動作について述べます。登録動作は gbsp のみならず LANDSCAPE サーバの他のエージェントも係ってきます。とりあえずこの節では各エージェントの役割には特に触れず全体の動きに着目した説明を行います。次節にて各エージェント役割を追いながら登録動作について説明を試みます。

gbsp を搭載した LANDSCAPE サーバに対しては、検索を始める最初の URL を与える必要があります。その方法には、二つあり、gbsp のセッティングにおいて、一つ URL を与えるという方法と、LANDSCAPE サーバにおいて既に用意されている、gbmp [UNDEF REF (gbmp)] からの情報で、LANDSCAPE サーバ上の位置を得ることができます。

まず LANDSCAPE は、この最初の URL を gbsp インデックス、root.queue.myurl に記録します。次にこの URL 中のリンクをたどり、次の URL を得、root.queue.myurl に記録します。最初の URL をホップ 0

とし、次に見つかった URL をホップ 1 として、ホップ数を同時に記録していきます。同じ URL にたどりつくは複数のパスが存在する可能性があります。その場合、より少ない方のホップ数を記録します。

LANDSCAPE はサーバ型 P2P システムを基本としていますので、複数の LANDSCAPE サーバが一つの URL ネットワークを検索、同じ URL をインデックス化する可能性があります。さらにそのインデックスをサーバ同士で交換します。そうした場合、見つけた URL を管理するサーバを決定するとき、よりホップ数が少ない方のサーバが担当するというルールになっています。このためホップ数とともに gbsp の識別子を記録し、一番ホップ数が少ない LANDSCAPE サーバから順番に決められた数の複数のサーバが担当となります。

root.queue.myurl に登録された URL は定期的アクセスされ、存在しているかのチェックのあと、URL の書誌情報、URL のコンテンツ内の地物の属性情報が URL の指し示す先から取得されます。取得された属性情報は、root.data の下に N グラム法によって分解された文字列として格納されます。URL の存在チェックが 5 回エラーとなると、この URL はインデックスから削除されます。

root.queue.myurl においては、サーバのアドレスも管理されます。URL から、プロトコル、サーバ名、ポート番号部分が抜き出され、URL の親ディレクトリとして登録されます。サーバ名は DNS によりアドレス変換され、プロトコル+アドレス+ポート番号 に変換され、root.addr の下に登録されます。root.addr は他の LANDSCAPE サーバからも情報を収集し、IP アドレスのキャッシュを構成します。IP アドレスは通し番号であることから、ここを検索すると、GB サーバや WWW サーバが世界にどのようなものがあるかがわかります。この機能を使い、すべての gbsp を持った LANDSCAPE サーバを検索し、クエリを投げることにより各種属性値の検索エンジンが実現します。

1.7 登録動作における各エージェントの役割

本節では、登録動作においてどのようなエージェントがどのような場面で動作するかについて述べる。まず最初に、存在するエージェントについて概略を述べる。

1. gbsp GB Spidering [?]

本マニュアルのエージェントである。スパイダリングに関係するインデックスを管理するエージェントである。インデックスに対する情報の登録、読み出し、タイマーによるトリガ命令をサポートしている。

2. gbmp GB Management Processor [UNDEF REF (gbmp)]

GLOBALBASE LANDSCAPE サーバに存在するコンテンツ、とくに map ファイルによる地図同士の接続に関して管理を行っているエージェント。存在するサーバ上のコンテンツすべてについて情報を持っているので、このエージェントが gbsp をトリガーすることによってコンテンツ収集が始まる。

3. gburl GB URL manager

収集されたコンテンツを、サーバ、ディレクトリ、リソースに分解し、gbsp を通じてインデックスに登録する役目を持つ、さらに、次に述べる gboar エージェントを起動する。

4. gboar GB Object Attribute Retriever

gburl によって起動され、各リソースの属性情報、および、リソースの含む地物の属性情報を収集し、インデックス化する。

5. gbaddr GB Address Management

サーバアドレスの管理を行うエージェントである。

次にこれらのエージェントの動きを説明します。

インデックスには 1.4 節で述べているように、あらかじめ決められたディレクトリが存在します。gbsp は起動すると、まずこのディレクトリが存在するかをチェックし、存在しない場合はこれらを生成する。次に各ディレクトリの管理エージェントを以下のように割り当てます。

- root.gbsp
- root.addr.gbaddr
- root.addr.ipv4.gbaddr
- root.addr.ipv6.gbaddr
- root.queue.gbsp
- root.queue.myurl.gburl
- root.queue.url.gburl
- root.data.gboar

以上のような初期化を行った後 gbsp は待機状態になります。LANDSCAPE 全体としての動きでは最初に、スパイダリングを開始する URL が必要です。一つの考え方は、gbmp が定期的に収集している URL からスパイダリングを開始するというものです。この場合、gbmp は、gbsp に対して、この URL に基づくディレクトリを root.queue.myurl の下に SPcreatePath (2.3.4 節) により生成しようとしています。root.queue.myurl の管理エージェントは gburl であり gbmp ではないので、gbsp は直接は生成はせず、管理エージェントの gburl に生成要求があったことを伝えます。

gburl はこれを受けとり、URL に対応するディレクトリを生成するまえに、URL よりプロトコル+サーバ+ポートのセットを生成し、この三つの組み合わせさせたサーバに対応するディレクトリを生成し、その下に、URL のディレクトリを生成します。これらのディレクトリの管理エージェントおよびトリガー・エージェントとして自分自身 gburl とする一方で、リソースにあたるにはトリガー・エージェントとして gboar を設定します。これにより、定期的に URL は gburl および gboar によってチェックされ、もし、url の先のリソースが消されている場合は、インデックスの対応するディレクトリも消されます。

一方、DNS によりサーバの名前よりアドレスを引き、それを、SPcreatePath (2.3.4 節) により、root.addr の下への登録を試みます。同様に gbsp はこれを保留し、エージェント gbaddr へ伝えます。

1.8 gburl の動き

gburl の詳しい動作は [UNDEF REF (gburl)] で行います。スパイダリングのための動作の概要について述べます。gburl は SPcreatePath トリガーを受信すると、自分の担当の path か、正しい path かをチェックします。そして、正しいと判断されれば、新しい path を生成します。あたらし path は自分自身を管理エージェントとし、また同時にトリガー・エージェントとしてタイマーを設定します。タイマーは初期値 0 秒、インターバルは所定のインターバル値が設定されます。

gburl にタイマー・トリガーがかかった場合、gburl はトリガーの発生した path について path の指し示す URL にはリソースが存在するかをチェックします。存在しないとなると、存在していない時間情報をカウントアップします。この情報がタイムアウトを超えていると、gburl はこの path を削除します。

path が存在している場合、かつ、オブジェクトリソース (.lst, .vet リソース) をさしている場合、gboar を起動します。gboar には、path をパラメータにエージェント起動トリガーを送ります。gboar は path に対応する URL のオブジェクトを取得し、オブジェクト中の属性を、root.data の下にインデックス化します。

一方、gburl は取得した path の対象となる URL にリンクされているマッピング (.map リソース)、またその先の座標系 (.crd リソース)、オブジェクトリソース (.lst,.vct リソース) を検索し、root.queue.myurl に登録する。もし、既に登録されている URL で会った場合は、ホップ数をチェックし、ホップ数を満たすものであれば、当該エージェントの動作するサーバアドレスとともにホップ数を記録する。ホップ数が既に登録されているホップ数より上回った場合は、特になにもしません。

また gburl は得られたパスの対応する URL の示すサーバの IP アドレスを解決する。この IP アドレスとともに、gbaddr エージェントを起動するトリガーを送ります。

1.9 gboar の動き

gboar は.lst,.vct リソースの管理対象とするオブジェクトの中身をすべて root.data の下にインデックス化する。オブジェクトの種類は多種多様なので、現在サポートしているオブジェクトの種類はこの gboar で処理するが、その他のオブジェクトの種類インデックス化が必要になった場合は、対応する別のエージェントを起動することもある。

gboar はオブジェクトの中の属性を呼び出している間に、URL を発見した場合、この URL を rootl.queue.myurl に登録する。結果的に gburl が起動される。

1.10 gbaddr の動き

エージェント gbaddr は root.myurl 配下に記録された URL より IP アドレスを取得し、root.addr 以下に記録して行く。

第2章 基本オペレーション

2.1 概要

gbsp に対する基本オペレーションの解説を行います。

2.2 継承情報

「xl(standard) エージェント・リファレンス・マニュアル [2]」の情報を継承しています。こちらも参照してください。

2.3 関数

2.3.1 SPdatabasePath

プロトタイプ

```
<SPdatabasePath> path </SPdatabasePath>
```

引数

path [1] 「xl(standard) エージェント・リファレンス・マニュアル」 [2] の「XLT_STRING(文字列型)」
オープンするインデックスファイルへのパス名

属性

評価形式

Applicative

所属エージェント

gbsp

所属環境

Env1

説明

path で指定するファイルを gbsp インデックスファイルとして指定し、オープンする。もしファイルが存在しない場合は、新しいものを作成する。

戻り値

正常終了時 「xl(standard) エージェント・リファレンス・マニュアル」 [2] の「XLT_NULL(ヌル型)」

エラー

- 「xl(standard) エージェント・リファレンス・マニュアル」 [2] の「XLE_SEMANTICS_TYPE_MISMATCH」
path の型があっていない。
- 「xl(standard) エージェント・リファレンス・マニュアル」 [2] の「XLE_PROTO_INV_PARAM」
ファイルが開けなかったか、ファイルが破壊されている。

参考

バグ

2.3.2 SPdefineAgent

プロトタイプ

```
<SPdefineAgent> name core-binary loading-file </SPdefineAgent>
```

引数

name [1] 「xl(standard) エージェント・リファレンス・マニュアル」 [2] の「XLT_STRING(文字列型)」
エージェント名

core-binary [1] 「xl(standard) エージェント・リファレンス・マニュアル」 [2] の「XLT_STRING(文字列型)」 エージェントの実行ファイル名

loading-file

[1] 「xl(standard) エージェント・リファレンス・マニュアル」 [2] の「XLT_STRING(文字列型)」 エージェント起動時の実行スクリプト

属性

評価形式

Applicative

所属エージェント

gbsp

所属環境

Env1

説明

エージェント名 *name* に具体的な実行ファイルと実行スクリプトを割り当てる。与えられたエージェント名は、コマンドラインで、

core-binary loading-file

を実行した場合と同じ機能となります。このエージェントの定義は gbsp の中でのみ通用するもので、他の場面では有効性をもちません。

戻り値

正常終了時 「xl(standard) エージェント・リファレンス・マニュアル」 [2] の「XLT_NULL(ヌル型)」

エラー

- 「xl(standard) エージェント・リファレンス・マニュアル」 [2] の「XLE_SEMANTICS_TYPE_MISMATCH」
name, core-binary, loading-file の型があていない。
- 「xl(standard) エージェント・リファレンス・マニュアル」 [2] の「XLE_PROTO_INV_PARAM」
エージェントの登録に失敗した。

参考

バグ

2.3.3 SPcd

プロトタイプ

<SPcd> *path* </SPcd>

引数

path [1] 「xl(standard) エージェント・リファレンス・マニュアル」 [2] の「XLT_STRING(文字列型)」
変更先ディレクトリパス

属性

評価形式

Applicative

所属エージェント

gbsp

所属環境

Env0

説明

カレントディレクトリを *path* に変更する。親ディレクトリへ変更する場合は、「..」を指定することができる。

戻り値

正常終了時 「xl(standard) エージェント・リファレンス・マニュアル」 [2] の「XLT_NULL(ヌル型)」

エラー

- 「xl(standard) エージェント・リファレンス・マニュアル」 [2] の「XLE_SEMANTICS_TYPE_MISMATCH」
path の型があていない。
- 「xl(standard) エージェント・リファレンス・マニュアル」 [2] の「XLE_PROTO_INV_PARAM」
path で指定されるディレクトリが存在しない。

参考

バグ

2.3.4 SPcreatePath

プロトタイプ

<SPcreatePath> *path* </SPcreatePath>

引数

path [1] 「xl(standard) エージェント・リファレンス・マニュアル」 [2] の「XLT_STRING(文字列型)」

ディレクトリパス

属性

評価形式

Applicative

所属エージェント

gbsp

所属環境

Env1

説明

path で指定されるディレクトリを生成する。この命令を発行したエージェントは、*path* の属する一番内側（パス名が一番長い）ディレクトリの管理エージェントの一つである必要がある。生成されたディレクトリの管理エージェントとして、この命令を発行したエージェントが設定される。

戻り値

正常終了時 「xl(standard) エージェント・リファレンス・マニュアル」 [2] の「XLT_NULL(ヌル型)」

エラー

- 「xl(standard) エージェント・リファレンス・マニュアル」 [2] の「XLE_SEMANTICS_TYPE_MISMATCH」
path の型があっていない。
- 「xl(standard) エージェント・リファレンス・マニュアル」 [2] の「XLE_PROTO_DUP_DEFINITION」
path で指定されるディレクトリが既に存在する。

参考

バグ

2.3.5 SPdeletePath

プロトタイプ

<SPdeletePath> *path* </SPdeletePath>

引数

path [1] 「xl(standard) エージェント・リファレンス・マニュアル」 [2] の「XLT_STRING(文字列型)」

ディレクトリパス

属性

評価形式

Applicative

所属エージェント

gbsp

所属環境

Env1

説明

path で指定されるディレクトリを削除する。 *path* の管理エージェント以外のエージェントが当該命令を発行すると権限エラーが発生する。

戻り値

正常終了時 「xl(standard) エージェント・リファレンス・マニュアル」 [2] の「XLT_NULL(ヌル型)」

エラー

- 「xl(standard) エージェント・リファレンス・マニュアル」 [2] の「XLE_SEMANTICS_TYPE_MISMATCH」
path の型があっていない。
- 「xl(standard) エージェント・リファレンス・マニュアル」 [2] の「XLE_PROTO_UNDEF_RESOURCE」
path で指定されるディレクトリが存在しない。

参考

バグ

2.3.6 SPls

プロトタイプ

- (1) <SPls> *path* </SPls>
- (2) <SPls/>

引数

path [0-1] 「xl(standard) エージェント・リファレンス・マニュアル」 [2] の「XLT_STRING(文字列型)」
ディレクトリパス

属性

評価形式

Applicative

所属エージェント

gbsp

所属環境

Env0

説明

path で指定されるディレクトリの直下のディレクトリをリスト化する。省略するとカレントディレクトリの直下のディレクトリをリスト化する。

戻り値

「xl(standard) エージェント・リファレンス・マニュアル」 [2] の「XLT_PAIR(ペア型、リスト型)」ディレクトリ名のリスト。

与えられた *path* が存在しない場合は全ディレクトリのリスト。

エラー

参考

バグ

2.3.7 SPpermitAgent

プロトタイプ

- (1) <SPpermitAgent> *path agent* </SPpermitAgent>
- (2) <SPpermitAgent> *path agent info* </SPpermitAgent>

引数

path [1] 「xl(standard) エージェント・リファレンス・マニュアル」 [2] の「XLT_STRING(文字列型)」ディレクトリパス

agent [1] 「xl(standard) エージェント・リファレンス・マニュアル」 [2] の「XLT_STRING(文字列型)」エージェント名

info [0-1] 「xl(standard) エージェント・リファレンス・マニュアル」 [2] の「XLT_RAW(RAW データ型)」バイナリ情報

属性

評価形式

Applicative

所属エージェント

gbsp

所属環境

Env1

説明

path で指定されるディレクトリの管理エージェントとして、*agent* を登録する。登録時、エージェント情報としてバイナリデータ *info* を指定することが出来る。*info* は省略可。

戻り値

正常終了時には「xl(standard) エージェント・リファレンス・マニュアル」 [2] の「XLT_NULL(ヌル型)」

エラー

- 「xl(standard) エージェント・リファレンス・マニュアル」 [2] の「XLE_SEMANTICS_TYPE_MISMATCH」
path, *agent*, *info* の型があていない。
- 「xl(standard) エージェント・リファレンス・マニュアル」 [2] の「XLE_PROTO_DUP_DEFINITION」
同一エージェントが既に管理エージェントとして登録されている。

参考

バグ

2.3.8 SPforbitAgent

プロトタイプ

<SPforbitAgent> *path agent* </SPforbitAgent>

引数

path [1] 「xl(standard) エージェント・リファレンス・マニュアル」 [2] の「XLT_STRING(文字列型)」
ディレクトリパス

agent [1] 「xl(standard) エージェント・リファレンス・マニュアル」 [2] の「XLT_STRING(文字列型)」
エージェント名

属性

評価形式

Applicative

所属エージェント

gbsp

所属環境

Env1

説明

path で指定されるディレクトリの管理エージェント *agent* を削除する。

戻り値

正常終了時には 「xl(standard) エージェント・リファレンス・マニュアル」 [2] の「XLT_NULL(ヌル型)」

エラー

- 「xl(standard) エージェント・リファレンス・マニュアル」 [2] の「XLE_SEMANTICS_TYPE_MISMATCH」
path, *agent* の型があていない。

参考

バグ

関連図書

- [1] 森洋久. LANDSCAPE スタートアップ・マニュアル. GLOBALBASE PROJECT, 2006.
- [2] 森洋久. xl(standard) エージェント・リファレンス・マニュアル. GLOBALBASE PROJECT, 2006.

履歴

1. 日時: 2010-12-01

マニュアル生成。(2010-12-01 版)

2. 日時: 2008-06-20

著者: 森 洋久 反映されたバージョン: ver.B.b17.04

このマニュアルを作成。ver.B.b17.04 よりこのマニュアルをアップしますが、実際は gbsp はまだ動作しません。