CHAPTER 3

ブロッキング API

この章では、ブロッキング API の機能と動作について説明し、コードの例を紹介します。また、ブロッキング API 固有の機能である応答タイムアウトについても説明します。

（注） *自動統合*の開発だけが必要な場合は、この章は飛ばして第 4 章「ノンブロッキング API」へ進んでください。

マルチスレッドのサポート

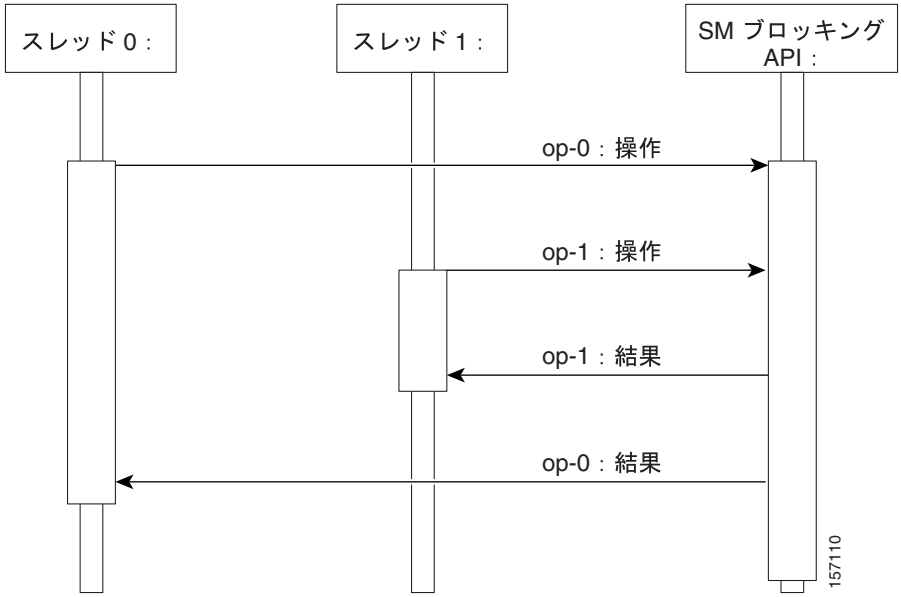
ブロッキング API では、メソッドを同時に呼び出すスレッドの数を設定できます。スレッド数の設定に関する詳細は、「C++ init メソッド」（p.3-32）を参照してください。

（注）　ブロッキング API でマルチスレッドにする場合、呼び出しの順序は保証**されません**。

マルチスレッドのサポートの例

スレッド -0 はタイム -0 で操作 -0 を呼び出し、スレッド -1 はタイム -1 で操作 -1 を呼び出します。このときタイム -1 はタイム -0 よりあとです。この例では、次の図（縦方向が時間）のように操作 -1 が操作 -0 よりも**前に**実行される可能性があります。

図 3-1　マルチスレッドのサポート



SM は 5 つのスレッドを割り当ててそれぞれの API インスタンスを処理します。複数のスレッドを持つ API を使用するマルチスレッド アプリケーションは、5 つのスレッド順に開発することを推奨します。より多くのスレッドを実装すると、呼び出しスレッドの遅延時間が長くなる可能性もあります。

操作タイムアウト エラー コード

ブロッキング操作は、SM から操作結果が取得された場合にだけ戻ります。ネットワークの誤作動やその他のエラーによって操作結果を取得できない場合、呼び出し側はいつまでも待機することになります。ただし SM API には、応答タイムアウト機能を使用して、こうした状況を避ける方法が用意されています。

呼び出し側は応答タイムアウト機能、すなわち **setReplyTimeout** メソッドを使用してタイムアウトを設定できます。タイムアウト期間内に応答が戻らない場合は、**ERROR_CODE_CLIENT_OPERATION_TIMEOUT** エラーで ReturnCode が戻ります。

応答タイムアウトを設定するには、**int** 値を指定して **setReplyTimeout** 機能を呼び出します。応答タイムアウトはミリ秒単位です。ゼロの値を指定すると、結果が届くまで操作が待機（フリーズまたは停止）状態になります。つまり、結果が届かなければ永久に待機状態のままになります。

ブロッキング API のメソッド

ここでは、ブロッキング API のメソッドについて説明します。メソッドの構文のあとに、各メソッド入力パラメータと戻り値の説明を示します。

ブロッキング API は、ノンブロッキング API の上位集合です。戻り値と結果の処理が異なる点を除けば、同じ操作の関数と構文の構造はどちらの API でも同じです。

C と C++ の API は、同じ関数シグニチャを共有していますが異なる点もあります。たとえばブロッキング C API では、関数名にはすべて **SMB_ プレフィクス**が付き、すべての関数の最初のパラメータは **SMB_HANDLE** 型の API ハンドルです。2 つの API のその他の相違点については、関数の説明で取り上げます。

ブロッキング API のサブスクライバ管理メソッドは次のカテゴリに分類されます。

- **IP およびプロパティの動的割り当て** — AAA システムとの統合に SM API を使用する場合は、次のようなメソッドを使用します。
  - login（p.3-5）
  - logoutByName（p.3-8）
  - logoutByNameFromDomain（p.3-10）
  - logoutByMapping（p.3-11）
  - loginCable（p.3-13）
  - logoutCable（p.3-15）

（注） これらのメソッドは、データベースに対するサブスクライバの追加や削除ではなく、既存のサブスクライバの動的パラメータ（IP アドレスなど）の変更に使用されます。

- **静的 / 手動のサブスクライバ設定** — たとえば GUI を利用する場合は、次のメソッドを使用します。
  - addSubscriber（p.3-16）
  - removeSubscriber（p.3-18）
  - removeAllSubscribers（p.3-19）
  - setPropertiesToDefault（p.3-29）
  - removeCustomProperties（p.3-30）

- サブスクライバ認識モードで個別に実行する単純な読み取りのみの操作は、次のメソッドを使用します。
  - getNumberOfSubscribers（p.3-19）
  - getNumberOfSubscribersInDomain（p.3-20）
  - getSubscriber（p.3-21）
  - subscriberExists（p.3-22）
  - subscriberLoggedIn（p.3-23）
  - getSubscriberNameByMapping（p.3-24）
  - getSubscriberNames（p.3-25）
  - getSubscriberNamesInDomain（p.3-26）
  - getSubscriberNamesWithPrefix（p.3-27）
  - getSubscriberNamesWithSuffix（p.3-28）
  - getDomains（p.3-29）

異なるカテゴリのメソッドを 1 つのアプリケーションに組み合わせて使用できます。この分類は、説明の目的のみに示したものです。

- **API のメンテナンス、初期化、接続、切断に使用されるメソッド**
  - C++ setLogger メソッド（p.3-31）
  - C++ init メソッド（p.3-32）
  - C SMB_init 関数（p.3-33）
  - C SMB_release 関数（p.3-34）
  - setReconnectTimeout（p.3-34）
  - setName（p.3-35）
  - connect（p.3-35）
  - disconnect（p.3-36）
  - isConnected（p.3-36）

（注） メソッドの説明の最後に記載されている例は、C++ の例です。メソッドの最後に記載されている例にはそれぞれ、その前に次のようなサンプル コードが記されています。

```
SmApiBlocking bapi;
// Init with default parameters
bapi.init();
// Connect to the SM
bapi.connect((char*)"1.1.1.1");
```

- login（p.3-5）
- logoutByName（p.3-8）
- logoutByNameFromDomain（p.3-10）
- logoutByMapping（p.3-11）
- loginCable（p.3-13）
- logoutCable（p.3-15）
- addSubscriber（p.3-16）
- removeSubscriber（p.3-18）
- removeAllSubscribers（p.3-19）

login

構文

```
ReturnCode* login(char* argName,
            char** argMappings,
            MappingType* argMappingTypes,
            int argMappingsSize,
            char** argPropertyKeys,
            char** argPropertyValues,
            int argPropertySize,
            char* argDomain,
            bool argIsAdditive,
            int argAutoLogoutTime)
```

説明

login メソッドは、SM データベース内の既存のサブスクライバのドメイン、マッピング、プロパティの追加や変更を実行します。このメソッドは部分データで呼び出すことができます。たとえば、マッピングだけ、またはプロパティだけを指定したり、未変更フィールドに NULL を入力することが可能です。

同じドメイン内に同じ（つまり衝突する）マッピングを持つ別のサブスクライバがすでに存在している場合、既存のサブスクライバから衝突しているマッピングが削除され、新しいサブスクライバにそのマッピングが割り当てられます。

指定したサブスクライバが SM データベース内に存在しない場合は、与えられたデータでサブスクライバが作成されます。

パラメータ

argName — 「サブスクライバ名のフォーマット」（p.2-9）を参照してください。

argMappings — 「ネットワーク ID マッピングについて」（p.2-10）のマッピングおよびマッピングの型についての説明を参照してください。マッピングが指定されておらず **argIsAdditive** フラグが TRUE の場合、直前のマッピングが保持されます。直前のマッピングが存在しない場合、操作はエラーとなります。

argMappingTypes — 「ネットワーク ID マッピングについて」（p.2-10）のマッピングおよびマッピングの型についての説明を参照してください。

argMappingsSize — argMappings 配列、および **argMappingTypes** 配列のサイズです。

argPropertyKeys — 「サブスクライバ プロパティ」（p.2-12）のプロパティのキーと値に関する説明を参照してください。

argPropertyValues — 「サブスクライバ プロパティ」（p.2-12）のプロパティのキーと値に関する説明を参照してください。

argPropertySize — argPropertyKeys 配列、および **argPropertyValues** 配列のサイズです。

argDomain — 「サブスクライバ ドメイン」（p.2-11）を参照してください。

ドメインが NULL であっても、そのサブスクライバにすでにドメインがある場合は、既存のドメインが保持されます。

ドメインが、サブスクライバにすでに割り当てられているドメインとは異なる場合、既存のドメインの SCE からサブスクライバが自動的に削除され、新しいドメインの SCE に移されます。

ArgIsAdditive — マッピング パラメータを参照してください。

- TRUE — この呼び出しによって与えられたマッピングをサブスクライバ レコードに追加します。
- FALSE — この呼び出しで与えられたマッピングで、サブスクライバ レコードの既存のマッピングを上書きします。

argAutoLogoutTime — このメソッドに引数として与えられたマッピングのみに適用されます。

- 正の値（N）— N 秒後に自動的にマッピングからログアウトします（logout メソッドの呼び出しに似ています）。
- 0 の値 — 指定されたマッピングのその時点での有効時間が維持されます。
- 負の値 — 指定されたマッピングに設定されている有効時間を無効にします。

戻り値

エラーが発生していない場合は、void 型の **ReturnCode** 構造体へのポインタが戻ります。

エラー コード

このメソッドが戻す可能性があるエラー コードは次のとおりです。

- ERROR_CODE_ILLEGAL_SUBSCRIBER_NAME
- ERROR_CODE_BAD_SUBSCRIBER_MAPPING
- ERROR_CODE_SUBSCRIBER_DOMAIN_ASSOCIATION
- ERROR_CODE_DATABASE_EXCEPTION
- ERROR_CODE_UNKNOWN

  このエラーは、次の場合に発生する可能性があります。

  - 存在しないサブスクライバまたはドメインのないサブスクライバの **domain** パラメータが NULL 値の場合
  - **propertyValues** パラメータの値が無効である場合

エラー コードの詳細については、「エラー コードのリスト」（p.A-1）を参照してください。

例

john という名前の既存のサブスクライバに IP アドレス 192.168.12.5 を追加し、マッピングは変更しない場合は、次のようにします。

```
MappingType map_type = IP_RANGE;
char* ip_address = "192.168.12.5";
bapi.login(
"john",                          // subscriber name(
&ip_address,
&map_type,
1,                               // one mapping
NULL, NULL, 0,                   // no properties
"subscribers",                   // domain
true,                            // isMappingAdditive is true
-1);                             // autoLogoutTime set to infinite
```

IP アドレス 192.168.12.5 を追加して、直前のマッピングに上書きする場合は、次のようにします。

```
MappingType map_type = IP_RANGE;
char* ip_address = "192.168.12.5";
bapi.login(
"john",                          // subscriber name(
&ip_address,
&map_type,
1,
NULL, NULL,0,
"subscribers",                   // domain
false,                           // isMappingAdditive is false
-1);                             // autoLogoutTime set to infinite
```

john に以前に割り当てられた 192.168.12.5 の自動ログアウト時間を延長するには、次のようにします。

```
MappingType map_type = IP_RANGE;
char* ip_address = "192.168.12.5";
bapi.login(
"john",                         // subscriber name(
&ip_address,
&map_type,   1,
NULL, NULL, 0,
"subscribers",                  // domain
false,                          // isMappingAdditive is false
300);                           // autoLogoutTime set to 300 seconds
```

john の動的プロパティ（package ID など）を変更するには、次のようにします。

```
char* prop_name = "packageID";
char* prop_value = "10";
bapi.login(
"john",
NULL, NULL, 0,
&prop_name,                     // property key
&prop_value,                    // property value
1,                              // one property
"subscribers",                  // domain
false,
-1);
```

john という名前の既存のサブスクライバに IP アドレス 192.168.12.5 を追加し、既存のマッピングは変更せず、*john* の動的プロパティ（package ID など）を変更する場合は、次のようにします。

```
MappingType map_type = IP_RANGE;
char* ip_address = "192.168.12.5";
char* prop_name = "packageID";
char* prop_value = "10";
bapi.login(
"john",
&ip_address,
&map_type,
1,
&prop_name,                     // property key
&prop_value,                    // property value
1,
"subscribers",                  // domain
true,                           // isMappingAdditive is set to true
-1);
```

logoutByName

構文

```
ReturnCode* logoutByName(char* argName,
                                char** argMappings,
                                MappingType* argMappingTypes,
                                int argMappingsSize)
```

説明

データベース内のサブスクライバを探し、そのサブスクライバからマッピングを削除します。

パラメータ

argName — 「サブスクライバ名のフォーマット」（p.2-9）を参照してください。

argMappings — 「ネットワーク ID マッピングについて」（p.2-10）のマッピングおよびマッピングの型についての説明を参照してください。マッピングが指定されていない場合は、すべてのサブスクライバ マッピングが削除されます。

argMappingTypes — 「ネットワーク ID マッピングについて」（p.2-10）のマッピングおよびマッピングの型についての説明を参照してください。

argMappingsSize — argMappings 配列、および **argMappingTypes** 配列のサイズです。

戻り値

ブール型の **ReturnCode** 構造体へのポインタ

- TRUE — サブスクライバが見つかり、そのサブスクライバのマッピングがサブスクライバ データベースから削除された場合
- FALSE — サブスクライバ データベースでサブスクライバが見つからなかった場合

エラー コード

このメソッドが戻す可能性があるエラー コードは次のとおりです。

- ERROR_CODE_SUBSCRIBER_DOES_NOT_EXIST
- ERROR_CODE _BAD_SUBSCRIBER_MAPPING
- ERROR_CODE_SUBSCRIBER_DOMAIN_ASSOCIATION
- ERROR_CODE_DOMAIN_NOT_FOUND
- ERROR_CODE_NOT_A_SUBSCRIBER_DOMAIN
- ERROR_CODE_DATABASE_EXCEPTION

エラー コードの詳細については、「エラー コードのリスト」（p.A-1）を参照してください。

例

サブスクライバ *john* の IP アドレス 192.168.12.5 を削除するには、次のようにします。

```
MappingType map_type = IP_RANGE;
char* ip_address = "192.168.12.5";
bapi.logoutByName(
"john",&ip_address,
&map_type,
1);
```

サブスクライバ *john* の IP アドレスをすべて削除するには、次のようにします。

```
bapi.logoutByName("john", NULL, NULL,0);
```

logoutByNameFromDomain

構文

```
ReturnCode* logoutByNameFromDomain (char* argName,
                                    char** argMappings,
                                    MappingType* argMappingTypes,
                                    int argMappingsSize,
                                    char* argDomain)
```

説明

指定されたドメインに基づいてデータベース内でサブスクライバを探し、そのサブスクライバからマッピングを削除します。

パラメータ

argName — 「サブスクライバ名のフォーマット」（p.2-9）を参照してください。

argMappings — 「ネットワーク ID マッピングについて」（p.2-10）のマッピングおよびマッピングの型についての説明を参照してください。マッピングが指定されていない場合は、すべてのサブスクライバ マッピングが削除されます。

argMappingTypes — 「ネットワーク ID マッピングについて」（p.2-10）のマッピングおよびマッピングの型についての説明を参照してください。

argMappingsSize — argMappings 配列、および **argMappingTypes** 配列のサイズです。

argDomain — 「サブスクライバ ドメイン」（p.2-11）を参照してください。

次の条件の*いずれか*に該当する場合、操作はエラーとなります。

- ドメインがヌルであるが、そのサブスクライバがデータベース内に存在し、ドメインに所属している場合
- 指定されたドメインが間違っている場合

戻り値

ブール型の **ReturnCode** 構造体へのポインタ

- TRUE — サブスクライバが見つかり、そのサブスクライバのマッピングがサブスクライバ データベースから削除された場合
- FALSE — サブスクライバ データベースでサブスクライバが見つからなかった場合

エラー コード

このメソッドが戻す可能性があるエラー コードは次のとおりです。

- ERROR_CODE_SUBSCRIBER_DOES_NOT_EXIST
- ERROR_CODE _BAD_SUBSCRIBER_MAPPING
- ERROR_CODE_SUBSCRIBER_DOMAIN_ASSOCIATION
- ERROR_CODE_DOMAIN_NOT_FOUND
- ERROR_CODE_NOT_A_SUBSCRIBER_DOMAIN
- ERROR_CODE_DATABASE_EXCEPTION

エラー コードの詳細については、「エラー コードのリスト」（p.A-1）を参照してください。

例

ドメイン *subscribers* から、サブスクライバ *john* の IP アドレス 192.168.12.5 を削除するには、次のようにします。

```
MappingType map_type = IP_RANGE;
char* ip_address = "192.168.12.5";
bapi.logoutByNameFromDomain(
"john",
&ip_address,
&map_type,
1,
"subscribers");
```

ドメイン *subscribers* から、サブスクライバ *john* の IP アドレスをすべてを削除するには、次のようにします。

```
bapi.logoutByNameFromDomain(
"john",
NULL,
NULL,
0,
"subscribers");
```

logoutByMapping

構文

```
ReturnCode* logoutByMapping( char* argMapping,
                  MappingType argMappingType,
                  char* argDomain)
```

説明

ドメインおよびマッピングに基づくサブスクライバを探し、そのサブスクライバ マッピングを削除します。サブスクライバはデータベースに残ります。

パラメータ

argMapping — 「ネットワーク ID マッピングについて」（p.2-10）のマッピングおよびマッピングの型についての説明を参照してください。

argMappingType — 「ネットワーク ID マッピングについて」（p.2-10）のマッピングおよびマッピングの型についての説明を参照してください。

argDomain — logoutByNameFromDomain メソッドの「パラメータ」（p.3-10）の説明を参照してください。

戻り値

ブール型の **ReturnCode** 構造体へのポインタ

- TRUE — サブスクライバが見つかり、サブスクライバ データベースから削除された場合
- FALSE — サブスクライバ データベースでサブスクライバが見つからなかった場合

エラー コード

このメソッドが戻す可能性があるエラー コードは次のとおりです。

- ERROR_CODE_SUBSCRIBER_DOES_NOT_EXIST
- ERROR_CODE _BAD_SUBSCRIBER_MAPPING
- ERROR_CODE_SUBSCRIBER_DOMAIN_ASSOCIATION
- ERROR_CODE_DOMAIN_NOT_FOUND
- ERROR_CODE_NOT_A_SUBSCRIBER_DOMAIN
- ERROR_CODE_DATABASE_EXCEPTION

エラー コードの詳細については、「エラー コードのリスト」（p.A-1）を参照してください。

例

ドメイン **subscribers** から IP アドレス 192.168.12.5 を削除するには、次のようにします。

```
bapi.logoutByMapping
("192.168.12.5",
IP_RANGE,
"subscribers");
```

loginCable

構文

```
ReturnCode* loginCable(char* argCpe,
                                    char* argCm,
                                    char* argIp,
                                    int argLease,
                                    char* argDomain,
                                    char** argPropertyKeys,
                                    char** argPropertyValues,
                                    int argPropertySize)
```

説明

ケーブル環境に適応した login メソッドで、SM 内でケーブル対応モジュールを呼び出します。このメソッドでは CPE が SM にログインされます。CM のログインには、CPE と CM の両方の引数に CM MAC アドレスを指定します。詳細は、『*Cisco SCMS Subscriber Manager User Guide*』の付録「Cable Environment」を参照してください。

（注）　SM データベース内の CPE の名前は、CPE と CM の値を **2** つの下線 [“_”] 文字で連結したものとなります。呼び出し側は CPE と CM の値の合計の長さが **38** 文字を超えていないことを確認する必要があります。

パラメータ

argCpe — CPE 固有の識別子（通常、MAC アドレス）

argCm — ケーブル モデム固有の識別子（通常、MAC アドレス）

argIp — CPE の IP アドレス

argLease — CPE のリース時間

argDomain — 「サブスクライバ ドメイン」（p.2-11）を参照してください。

通常は CMTS IP です。

（注）　CMTS IP がドメイン名として正しく解釈されるためには、SM にドメインの別名を設定する必要があります。別名の設定については、『*Cisco SCMS Subscriber Manager User Guide*』の「Configuring Domains」を参照してください。

argPropertyKeys —「サブスクライバ プロパティ」（p.2-12）のキーと値に関する説明を参照してください。CPE のアプリケーション プロパティが一部またはまったく設定されていない場合、不足しているアプリケーション プロパティの値はこの CPE が属する CM のアプリケーション プロパティからコピーされます。CM の各アプリケーション プロパティは、その CM に属している CPE のデフォルト値として使用されます。

argPropertyValues —「サブスクライバ プロパティ」（p.2-12）のキーと値に関する説明を参照してください。

argPropertySize — argPropertyKeys 配列、および **argPropertyValues** 配列のサイズです。

戻り値

void 型の **ReturnCode** 構造体へのポインタ

エラー コード

なし

例

CM1 という名前の CM に IP アドレス 192.168.12.5 を追加し、リース時間を 2 時間にするには、次のようにします。

```
bapi.loginCable
("CM1",
"CM1",
"192.168.12.5",
7200,          // lease time in seconds
"subscribers",
NULL, NULL, 0);           // no properties
```

CM1 に属する *CPE1* という名前の CPE に IP アドレス 192.168.12.50 を追加し、リース時間を 1 時間にするには、次のようにします。

```
bapi.loginCable(
"CPE1",
"CM1",
"192.168.12.50",
3600,      // lease time in seconds
"subscribers",
NULL, NULL, 0);
```

logoutCable

構文

```
ReturnCode* logoutCable(char* argCpe,
                                    char* argCm,
                                    char* argIp,
                                    char* argDomain)
```

説明

SM ケーブル対応モジュールにログアウト イベント（CPE のオフライン）を示します。

パラメータ

argCpe — loginCable メソッドの「パラメータ」（p.3-13）の説明を参照してください。

argCm — loginCable メソッドの「パラメータ」（p.3-13）の説明を参照してください。

argIp — loginCable メソッドの「パラメータ」（p.3-13）の説明を参照してください。

argDomain — loginCable メソッドの「パラメータ」（p.3-13）の説明を参照してください。

戻り値

ブール型の **ReturnCode** 構造体へのポインタ

- TRUE — その CPE が見つかり、サブスクライバ データベースから削除された場合
- FALSE — サブスクライバ データベースでその CPE が見つからなかった場合

エラー コード

なし

例

CM1 に属する *CPE1* から IP アドレス 192.168.12.5 を削除するには、次のようにします。

```
bool isExist = bapi.logoutCable
("CPE1",
"CM1",
"192.168.12.5",
"subscribers");
```

addSubscriber

構文

```
ReturnCode* addSubscriber( char* argName,
                                    char** argMappings,
                                    MappingType* argMappingTypes,
                                    int argMappingsSize,
                                    char** argPropertyKeys,
                                    char** argPropertyValues,
                                    int argPropertySize,
                                    char** argCustomPropertyKeys,
                                    char** argCustomPropertyValues,
                                    int argCustomPropertySize,
char* argDomain)
```

説明

与えられたデータに基づいて新しいサブスクライバ レコードを 1 つ作成し、これを SM データベースに追加します。この名前のサブスクライバがすでに存在する場合、そのサブスクライバは新しいサブスクライバが追加される前に削除されます。**login** では値が渡されたフィールドが変更され、未指定フィールドは変更されませんが、**addSubscriber** では、値が渡されたパラメータによって、サブスクライバが指定どおりに設定されます。

（注）　既存のサブスクライバには、**addSubscriber** ではなく **login** メソッドを呼び出すことを推奨します。動的なマッピングおよびプロパティは **login** を使用して設定する必要があります。静的なマッピングおよびプロパティは、**addSubscriber** を使用してサブスクライバの初回作成時に設定する必要があります。

（注）　**addSubscriber** を使用する場合、自動ログアウト機能は常にディセーブルです。自動ログアウトをイネーブルにするには、**login** を指定します。

例：

サブスクライバ データベース内にすでにセットアップされているサブスクライバ *AB* には、*IP1* という IP マッピングが 1 つあります。

AB に対する **addSubscriber** 操作がマッピングの指定なしに呼び出されると（**mappings** と **mappingTypes** のフィールドが NULL）、*AB* はマッピングがないままになります。

ただし、これらの NULL 値パラメータを使って **login** 操作を呼び出しても *AB* のマッピングは変更されず、*AB* は直前の IP マッピング *IP1* を保持したままになります。

パラメータ

argName — 「サブスクライバ名のフォーマット」（p.2-9）を参照してください。

argMappings — 「ネットワーク ID マッピングについて」（p.2-10）のマッピングおよびマッピングの型についての説明を参照してください。

argMappingTypes — 「ネットワーク ID マッピングについて」（p.2-10）のマッピングおよびマッピングの型についての説明を参照してください。

argMappingsSize — argMappings 配列、および **argMappingTypes** 配列のサイズです。

argPropertyKeys — 「サブスクライバ プロパティ」（p.2-12）のプロパティのキーと値に関する説明を参照してください。

argPropertyValues — 「サブスクライバ プロパティ」（p.2-12）のプロパティのキーと値に関する説明を参照してください。

argPropertySize — argPropertyKeys 配列、および **argPropertyValues** 配列のサイズです。

argCustomPropertyKeys — 「カスタム プロパティ」（p.2-12）のカスタム プロパティのキーと値に関する説明を参照してください。

argCustomPropertyValues — 「カスタム プロパティ」（p.2-12）のカスタム プロパティのキーと値に関する説明を参照してください。

argPropertySize — argCustomPropertyKeys 配列、および **argCustomPropertyValues** 配列のサイズです。

argDomain — 「サブスクライバ ドメイン」（p.2-11）を参照してください。

NULL 値は、サブスクライバがドメインレスであることを示します。

戻り値

void 型の **ReturnCode** 構造体へのポインタ

エラー コード

このメソッドが戻す可能性があるエラー コードは次のとおりです。

- ERROR_CODE_ILLEGAL_SUBSCRIBER_NAME
- ERROR_CODE _BAD_SUBSCRIBER_MAPPING
- ERROR_CODE_DOMAIN_NOT_FOUND
- ERROR_CODE_SUBSCRIBER_ALREADY_EXISTS
- ERROR_CODE_SUBSCRIBER_DOMAIN_ASSOCIATION
- ERROR_CODE_DATABASE_EXCEPTION
- ERROR_CODE_UNKNOWN

  このエラー コードは、**propertyValues** パラメータに無効な値が指定されたことを示します。

エラー コードの詳細については、「エラー コードのリスト」（p.A-1）を参照してください。

例

カスタム プロパティを指定して新しいサブスクライバ *john* を追加するには、次のようにします。

```
char* propKeys[] = { "work_phone", "home_phone" };
char *propValues[] = { "123456", "898765" };
bapi.addSubscriber
(
"john",
NULL, NULL, 0,// dynamic mappings will be set by login
NULL, NULL, 0,// dynamic properties will be set by login
propKeys, propValues, 2,// 2 custom properties
"subscribers");        // default domain
```

removeSubscriber

構文

```
ReturnCode* removeSubscriber(char* argName)
```

説明

SM データベースから 1 つのサブスクライバを完全に削除します。

パラメータ

argName — 「サブスクライバ名のフォーマット」（p.2-9）を参照してください。

戻り値

ブール型の **ReturnCode** 構造体へのポインタ

- TRUE — データベースでサブスクライバが見つかり、正常に削除された場合
- FALSE — TRUE 条件に合致しない場合。つまり、データベースでサブスクライバが見つからなかったか、または見つかったが正常に削除されなかった場合

エラー コード

このメソッドが戻す可能性があるエラー コードは次のとおりです。

- ERROR_CODE_ILLEGAL_SUBSCRIBER_NAME
- ERROR_CODE _SUBSCRIBER_DOES_NOT_EXIST
- ERROR_CODE_DATABASE_EXCEPTION

エラー コードの詳細については、「エラー コードのリスト」（p.A-1）を参照してください。

例

サブスクライバ *john* をデータベースから完全に削除するには、次のようにします。

```
bapi.removeSubscriber("john");
```

removeAllSubscribers

構文

```
ReturnCode* removeAllSubscribers()
```

説明

SM からすべてのサブスクライバを削除し、データベースにサブスクライバが 1 つもない状態にします。

（注）　このメソッドは実行に時間がかかる場合もあります。操作タイムアウトによる除外が発生しないようにするため、このメソッドを呼び出す前に操作タイムアウトを大きな値（最大 5 分）に設定してください。

戻り値

void 型の **ReturnCode** 構造体へのポインタ

エラー コード

なし

getNumberOfSubscribers

構文

```
ReturnCode* getNumberOfSubscribers()
```

説明

SM データベース内のサブスクライバの合計数を取得します。

戻り値

SM 内のサブスクライバ数を表す整数が格納された **ReturnCode** 構造体へのポインタ

エラー コード

なし

getNumberOfSubscribersInDomain

構文

```
ReturnCode* getNumberOfSubscribersInDomain(char* argDomain)
```

説明

ある 1 つのサブスクライバ ドメイン内のサブスクライバ数を取得します。

パラメータ

argDomain — SM のドメイン リポジトリにあるサブスクライバ ドメインの名前

戻り値

所定のドメイン内のサブスクライバ数を表す整数が格納された **ReturnCode** 構造体へのポインタ

エラー コード

このメソッドが戻す可能性があるエラー コードは次のとおりです。

- ERROR_CODE_NOT_A_SUBSCRIBER_DOMAIN
- ERROR_CODE _DOMAIN_NOT_FOUND

エラー コードの詳細については、「エラー コードのリスト」（p.A-1）を参照してください。

getSubscriber

構文

```
ReturnCode* getSubscriber(char* argName)
```

説明

サブスクライバ レコードを取得します。各フィールドは、整数、文字列、または文字列配列としてフォーマットされます。このメソッドの戻り値に関する説明を参照してください。SM データベース内にそのサブスクライバが存在しない場合は、例外処理が実行されます。

パラメータ

argName — 「サブスクライバ名のフォーマット」（p.2-9）を参照してください。

戻り値

9 つの要素を持つ **ReturnCode** 構造体の配列が格納された **ReturnCode** 構造体へのポインタ。NULL 値となる配列要素はありません。

要素の値と意味は次のとおりです。

インデックス 0	サブスクライバ名（**char***）
インデックス 1	マッピングの配列（**char****）
インデックス 2	マッピングの型の配列（**short***）
インデックス 3	ドメイン名（**char***）
インデックス 4	プロパティ名の配列（**char****）
インデックス 5	プロパティ値の配列（**char****）
インデックス 6	プロパティ名の配列（**char****）
インデックス 7	カスタム プロパティ値の配列（**char****）
インデックス 8	自動ログアウト時間（現在からの秒数）の配列、またはマッピング（インデックス 1）ごとに設定されていない場合（long 1*）は -1

エラー コード

このメソッドが戻す可能性があるエラー コードは次のとおりです。

- ERROR_CODE_SUBSCRIBER_DOES_NOT_EXIST
- ERROR_CODE_DATABASE_EXCEPTION

エラー コードの詳細については、「エラー コードのリスト」（p.A-1）を参照してください。

例

john のサブスクライバ レコードを取得するには、次のようにします。

```
ReturnCode* sub = bapi.getSubscriber("john");
// sub name
char* name = sub->u.objectArray[0]->u.stringVal;
// sub mapping
char** mappings = sub->u.objectArray[1]->u.stringArrayVal;
// mappings types
short* types = sub->u.objectArray[2]->u.shortArrayVal;
char* domainName = (char*)sub->u.objectArray[3]->u.stringVal;
char** propertyNames = (char**)sub->u.objectArray[4]->u.stringArrayVal;
char** propertyValues = (char**)sub->u.objectArray[5]->u.stringArrayVal;
char** customPropertyName = (char**)sub->u.objectArray[6]->u.stringArrayVal;
char** customPropertyValues = (char**)sub->u.objectArray[7]->u.stringArrayVal;
long* autoLogoutTime = sub->u.objectArray[8]->u.longArrayVal;
```

subscriberExists

構文

```
ReturnCode* subscriberExists(char* argName)
```

説明

あるサブスクライバが SM データベース内に存在していることを確認します。

パラメータ

argName — 「サブスクライバ名のフォーマット」（p.2-9）を参照してください。

戻り値

ブール型の ReturnCode 構造体へのポインタ

- TRUE — SM データベースでそのサブスクライバが見つかった場合
- FALSE — そのサブスクライバが見つからなかった場合

エラー コード

なし

subscriberLoggedIn

構文

```
ReturnCode* subscriberLoggedIn(char* argName)
```

説明

SM データベース内の既存のサブスクライバがログインしているかどうかを確認します。つまり、そのサブスクライバが SCE データベースにも存在しているかどうか確認します。

SM が *Pull* モードで機能するように設定されている場合、このメソッドで TRUE の値が戻っても、そのサブスクライバが実際に SCE データベース内に存在しているとは**限りません**。単に必要に応じて SCE がサブスクライバを pull できるようになっていることを意味します。

SM データベース内にそのサブスクライバが存在しない場合は、例外がスローされます。

パラメータ

argName — 「サブスクライバ名のフォーマット」（p.2-9）を参照してください。

戻り値

ブール型の **ReturnCode** 構造体へのポインタ

- TRUE — そのサブスクライバがログインしている場合
- FALSE — そのサブスクライバがログインしていない場合

エラー コード

このメソッドが戻す可能性があるエラー コードは次のとおりです。

- ERROR_CODE_ILLEGAL_SUBSCRIBER_NAME
- ERROR_CODE_DATABASE_EXCEPTION

エラー コードの詳細については、「エラー コードのリスト」（p.A-1）を参照してください。

getSubscriberNameByMapping

構文

```
ReturnCode* getSubscriberNameByMapping(char* argMapping,
MappingType argMappingType,
char* argDomain)
```

説明

マッピングおよびドメインに基づいてサブスクライバ名を検索します。

パラメータ

argMapping — 「ネットワーク ID マッピングについて」（p.2-10）のマッピングおよびマッピングの型についての説明を参照してください。

argMappingType — 「ネットワーク ID マッピングについて」（p.2-10）のマッピングおよびマッピングの型についての説明を参照してください。

argDomain — そのサブスクライバが属しているドメインの名前。次の条件のいずれかに該当する場合、操作はエラーとなります。

- ドメインがヌルであるが、そのサブスクライバがデータベース内に存在し、ドメインに所属している場合
- 指定されたドメインが間違っている場合

戻り値

文字列（**char***）型の **ReturnCode** 構造体へのポインタ

- サブスクライバ名 — サブスクライバ レコードが見つかった場合
- NULL — 与えられたマッピングを持つサブスクライバ レコードが SM データベース内で見つからなかった場合

エラー コード

このメソッドが戻す可能性があるエラー コードは次のとおりです。

- ERROR_CODE_DOMAIN_NOT_FOUND
- ERROR_CODE _BAD_SUBSCRIBER_MAPPING
- ERROR_CODE_NOT_A_SUBSCRIBER_DOMAIN
- ERROR_CODE_DATABASE_EXCEPTION

エラー コードの詳細については、「エラー コードのリスト」（p.A-1）を参照してください。

getSubscriberNames

構文

```
ReturnCode* getSubscriberNames(char* argFirstName,
int argAmount)
```

説明

SM データベースから、サブスクライバ名を一括して取得します。**argFirstName** から始まり、アルファベット順に **argAmount** の数だけサブスクライバが取得されます。

argFirstName が NULL の場合、SM データベース内に存在するサブスクライバ名のうちアルファベット順で最初のサブスクライバ名が使用されます。

（注）　**getNumOfSubscribers** からの戻り値が必ずサブスクライバの合計数（全バルク）であるとは**限りません**。たとえば、取得中にいくつかのサブスクライバの追加や削除が実行された場合は合計数と異なる可能性があります。

パラメータ

argFirstName — 最後のバルクの最後のサブスクライバ名（検索する最初の名前）。アルファベット順で最初のサブスクライバから開始する場合は、NULL を使用します。

argAmount — 取得するサブスクライバの数を限定します。この値が SM の制限値（1000）より大きい場合は、SM の制限値が使用されます。

（注）　このパラメータに 500 を超える値を指定することは**推奨しません**。

戻り値

アルファベット順のサブスクライバ名リストが格納された文字列配列型（**char****）の **ReturnCode** 構造体へのポインタ。

このメソッドは、要求されたサブスクライバから開始して、SM データベース内で見つかったすべてのサブスクライバを戻します。配列のサイズは、**argAmount** の最小値と SM 制限値（1000）によって制限されます。

エラー コード

このメソッドが戻す可能性があるエラー コードは次のとおりです。

- ERROR_CODE_ILLEGAL_SUBSCRIBER_NAME
- ERROR_CODE_DATABASE_EXCEPTION

エラー コードの詳細については、「エラー コードのリスト」（p.A-1）を参照してください。

例

サブスクライバ名をアルファベット順のリストで取得するには、次のようにします。

```
bool hasMoreSubscribers;
char* lastBulkEnd = NULL;
char tmpName[50];
int bulkSize = 100;
do
{
ReturnCode* subscribers =
smApi.getSubscriberNames(lastBulkEnd,bulkSize);
hasMoreSubscribers = false;
if ((isReturnCodeError(subscribers) == false) &&
(subscribers->type == STRING_ARRAY_T) &&(subscribers->u.stringArrayVal != NULL))
{
for (int i = 0; i <subscribers->size; i++)
{
// do something with subscribers->u.stringArrayVal[i]
}
if (subscribers->size == bulkSize)
{
hasMoreSubscribers = true;
strcpy (tmpName, subscribers->u.stringArrayVal[bulkSize - 1]);
lastBulkEnd = tmpName;
}
}
freeReturnCode(subscribers);
} while (hasMoreSubscribers);
```

getSubscriberNamesInDomain

構文

```
ReturnCode* getSubscriberNamesInDomain( char* argFirstName,
int argAmount,
char* argDomain)
```

説明

指定されたドメインに関連付けられているサブスクライバを SM データベースから取得します。

この操作の関数は、「getSubscriberNames」（p.3-25）操作の場合と同じです。

パラメータ

argFirstName — getSusbcriberNames 操作の「パラメータ」（p.3-25）の説明を参照してください。

argAmount — getSusbcriberNames 操作の「パラメータ」（p.3-25）の説明を参照してください。

argDomain — SM のドメイン リポジトリにあるサブスクライバ ドメインの名前

戻り値

指定されたドメインに属しているサブスクライバ名のアルファベット順配列。

詳細については、getSusbcriberNames 操作の「戻り値」（p.3-25）の説明を参照してください。

エラー コード

このメソッドが戻す可能性があるエラー コードは次のとおりです。

- ERROR_CODE_ILLEGAL_SUBSCRIBER_NAME
- ERROR_CODE _DOMAIN_NOT_FOUND
- ERROR_CODE_DATABASE_EXCEPTION

エラー コードの詳細については、「エラー コードのリスト」（p.A-1）を参照してください。

getSubscriberNamesWithPrefix

構文

```
ReturnCode* getSubscriberNamesWithPrefix(char* argFirstName,
int argAmount,
char* argPrefix)
```

説明

指定されたプレフィクスから名前が始まるサブスクライバを SM データベースから取得します。

この操作の関数は、「getSubscriberNames」（p.3-25）操作の場合と同じです。

パラメータ

argFirstName — getSusbcriberNames 操作の「パラメータ」（p.3-25）の説明を参照してください。

argAmount — getSusbcriberNames 操作の「パラメータ」（p.3-25）の説明を参照してください。

argPrefix — 要求対象のサブスクライバ名のプレフィックスを示す文字列（大文字と小文字が区別されます）。

戻り値

要求されたプレフィクスから始まるサブスクライバ名のアルファベット順配列。

詳細については、getSusbcriberNames 操作の「戻り値」（p.3-25）の説明を参照してください。

エラー コード

このメソッドが戻す可能性があるエラー コードは次のとおりです。

- ERROR_CODE_ILLEGAL_SUBSCRIBER_NAME
- ERROR_CODE_DATABASE_EXCEPTION

エラー コードの詳細については、「エラー コードのリスト」（p.A-1）を参照してください。

getSubscriberNamesWithSuffix

構文

```
ReturnCode* getSubscriberNamesWithSuffix(char* argFirstName,
int argAmount,
char* argSuffix)
```

説明

指定されたサフィックスが名前の最後に付いているサブスクライバを SM データベースから取得します。

この操作の関数は、「getSubscriberNames」（p.3-25）操作の場合と同じです。

パラメータ

argFirstName — getSusbcriberNames 操作の「パラメータ」（p.3-25）の説明を参照してください。

argAmount — getSusbcriberNames 操作の「パラメータ」（p.3-25）の説明を参照してください。

argSuffix — 要求対象のサブスクライバ名のサフィックスを示す文字列（大文字と小文字が区別されます）。

戻り値

要求されたサフィックスで終わるサブスクライバ名のアルファベット順配列。

詳細については、getSusbcriberNames 操作の「戻り値」（p.3-25）の説明を参照してください。

エラー コード

このメソッドが戻す可能性があるエラー コードは次のとおりです。

- ERROR_CODE_ILLEGAL_SUBSCRIBER_NAME
- ERROR_CODE_DATABASE_EXCEPTION

エラー コードの詳細については、「エラー コードのリスト」（p.A-1）を参照してください。

getDomains

構文

```
ReturnCode* getDomains()
```

説明

SM ドメイン リポジトリ内の現在のサブスクライバ ドメインのリストを提供します。

戻り値

SM 内のサブスクライバ ドメイン名の完全リストが格納された文字列配列型（**char****）の **ReturnCode** 構造体へのポインタ

エラー コード

なし

setPropertiesToDefault

構文

```
ReturnCode* setPropertiesToDefault(char* argName,
char** argPropertyKeys,
int argPropertySize)
```

説明

1 つのサブスクライバの指定されたアプリケーション プロパティをリセットします。アプリケーションがインストールされている場合、該当するアプリケーション プロパティは、その時点でロードされているアプリケーション情報に基づいて、そのプロパティのデフォルト値に設定されます。アプリケーションがインストールされていない場合は、エラー コード **ERROR_CODE_ILLEGAL_STATE** が戻ります。

パラメータ

argName — 「サブスクライバ名のフォーマット」（p.2-9）を参照してください。

argPropertyKeys — 「サブスクライバ プロパティ」（p.2-12）のプロパティのキーと値に関する説明を参照してください。

argPropertySize — argPropertyKeys 配列のサイズです。

戻り値

void 型の **ReturnCode** 構造体へのポインタ

エラー コード

このメソッドが戻す可能性があるエラー コードは次のとおりです。

- ERROR_CODE_ILLEGAL_SUBSCRIBER_NAME
- ERROR_CODE _BAD_SUBSCRIBER_MAPPING
- ERROR_CODE_DOMAIN_NOT_FOUND
- ERROR_CODE_SUBSCRIBER_DOES_NOT_EXIST
- ERROR_CODE_NOT_A_SUBSCRIBER_DOMAIN
- ERROR_CODE_DATABASE_EXCEPTION

エラー コードの詳細については、「エラー コードのリスト」（p.A-1）を参照してください。

removeCustomProperties

構文

```
ReturnCode* removeCustomProperties(char* argName,
char** argCustomPropertyKeys,
int argCustomPropertySize)
```

説明

1 つのサブスクライバの指定されたカスタム プロパティをリセットします。

パラメータ

argName — 「サブスクライバ名のフォーマット」（p.2-9）を参照してください。

argCustomPropertyKeys — 「カスタム プロパティ」（p.2-12）のカスタム プロパティのキーと値に関する説明を参照してください。

argCustomPropertySize — argCustomPropertyKeys 配列のサイズです。

戻り値

void 型の **ReturnCode** 構造体へのポインタ

エラー コード

このメソッドが戻す可能性があるエラー コードは次のとおりです。

- ERROR_CODE_ILLEGAL_SUBSCRIBER_NAME
- ERROR_CODE _SUBSCRIBER_DOES_NOT_EXIST
- ERROR_CODE_DATABASE_EXCEPTION

エラー コードの詳細については、「エラー コードのリスト」（p.A-1）を参照してください。

C++ setLogger メソッド

構文

```
void setLogger(Logger *argLogger)
```

説明

Logger 抽象クラスの実装を設定します。このメソッドは、SM API ログ メッセージをホスト アプリケーション ログに統合するために使用します。

パラメータ

argLogger — Logger 抽象クラスの実装です。

戻り値

なし

C++ init メソッド

構文

```
Bool init(int argSupportedThreads,
int argThreadPriority,
Uint32 argBufferSize,
Uint32 argKeepAliveDuration,
Uint32 argConnectionTimeout,
Uint32 argReconnectTimeout)
```

説明

API を設定し初期化します。

（注）　このメソッドは、C++ API の操作を実行する前に呼び出す必要があります。

パラメータ

argSupportedThreads — API がサポートするスレッド数

argThreadPriority — PRPC プロトコル ネットワーク スレッドのプライオリティ

argBufferSize — 内部バッファ サイズ（デフォルト値は 2,000,000 バイト）

argKeepAliveDuration — PRPC プロトコルのキープアライブ メッセージ間の必要な遅延に関するヒント（デフォルト値は 10 秒）

argConnectionTimeout — 応答のない PRPC プロトコル接続の必要なタイムアウトに関するヒント（デフォルト値は 20 秒）

argReconnectTimeout — SM への接続がダウンしたとき API が接続を再開しようとするまでのタイムアウト時間

戻り値

ブール値

- TRUE — 成功
- FALSE — 失敗

例

```
SmBlockingApi bapi;
bool success = bapi.init(10,
0,
2000000, //default
10,      //default
20,      //default
0);      //default (no reconnect)
```

C SMB_init 関数

構文

```
SMB_HANDLE SMB_init ( int argSupportedThreads,
int argThreadPriority,
Uint32 argBufferSize,
Uint32 argKeepAliveDuration,
Uint32 argConnectionTimeout)
```

説明

API の割り当て、設定、初期化を行います。

（注）　このメソッドは、C API の操作を実行する前に呼び出す必要があります。

パラメータ

argSupportedThreads — API がサポートするスレッド数

argThreadPriority — PRPC プロトコル ネットワーク スレッドのプライオリティ

argBufferSize — 内部バッファ サイズ（デフォルト値は 2,000,000 バイト）

argKeepAliveDuration — PRPC プロトコルのキープアライブ メッセージ間の必要な遅延に関するヒント（デフォルト値は 10 秒）

argConnectionTimeout — 応答のない PRPC プロトコル接続の必要なタイムアウトに関するヒント（デフォルト値は 20 秒）

戻り値

API への **SMB_HANDLE** ハンドル。このハンドルが NULL の場合、初期化は失敗しました。失敗していない場合は NULL 以外の値が戻ります。

例

```
SMB_HANDLE api;
// initialize an API
api = SMB_init(10, // 10 threads
0,
300000, // 3,000,000 bytes
10,     // default
30);    // 30 sec connection timeout
```

C SMB_release 関数

構文

```
void SMB_release(SMB_HANDLE argApiHandle)
```

説明

API で使用されたリソースを解放します。API を使用した場合、最後にこの関数を呼び出す必要があります。

パラメータ

argApiHandle — SMB_init 関数を使用して受け取った API ハンドル

戻り値

なし

setReconnectTimeout

構文

```
void setReconnectTimeout(Uint32 reconnectTimeout)
```

説明

再接続タイムアウトを設定します。

SM への接続がダウンしたとき API は「再接続タイムアウト」の秒数が過ぎると接続を再確立しようとします。

パラメータ

reconnectTimeout — タイムアウト

戻り値

なし

setName

構文

```
void setName(char *argName)
```

説明

API の名前を設定します。この名前はその API-SM 接続固有の識別子として使用されます。connect メソッドを呼び出す前に、setName 関数を呼び出す必要があります。

パラメータ

argName — API 名

戻り値

なし

connect

構文

```
bool connect(char* argHostName, Uint16 argPort = 14374)
```

説明

SM への PRPC プロトコル接続の確立を試行します。

パラメータ

argHostName — SM の IP アドレスまたはホスト名

argPort — SM に接続する TCP ポート（デフォルトは 14374）

戻り値

ブール値

- TRUE — 成功
- FALSE — 失敗

disconnect

構文

```
bool disconnect()
```

説明

SM への PRPC プロトコル接続の終了を試行します。

戻り値

ブール値

- TRUE — 成功
- FALSE — 失敗

isConnected

構文

```
bool isConnected();
```

説明

SM への PRPC プロトコル接続が確立され稼働中であるかどうかを確認します。

戻り値

ブール値

- TRUE — 接続が確立されています。
- FALSE — 接続は確立されていません。

ブロッキング API C++ コードの例

ここでは、次の 2 つのコード例を紹介します。

サブスクライバ数の取得

次の例では、SM データベース内のサブスクライバ総数と各サブスクライバ ドメイン内のサブスクライバ数を **stdout** に出力します。

```
include "SmApiBlocking.h"
include <stdio.h>
int main(int argc, char* argv[])
{
SmApiBlocking bapi;
//initiation
bapi.init();
bapi.setReplyTimeout(300000); //set timeout for 5 minutes
bapi.connect(argv[1]); // connect to the SM
//operations
ReturnCode* domains = bapi.getDomains();
ReturnCode* totalSubscribers=bapi.getNumberOfSubscribers();
if ((isReturnCodeError(domains) == false) &&
(isReturnCodeError(totalSubscribers) == false))
{
printf("number of susbcribers in the database:\t\t %d\n",
totalSubscribers->u.intVal);
for (int i=0; i<domains->size; i++)
{
ReturnCode* numberOfSusbcribersInDomain=
bapi.getNumberOfSubscribersInDomain(
domains->u.stringArrayVal[i]);
if (isReturnCodeError(numberOfSusbcribersInDomain) == false)
{
printf("number of susbcribers domain %s:\t\t%d\n",
domains->u.stringArrayVal[i],
numberOfSusbcribersInDomain->u.intVal);
}
freeReturnCode (numberOfSusbcribersInDomain);
}
}
freeReturnCode (domains);
freeReturnCode (totalSubscribers);
//finalization
bapi.disconnect();
return 0;
}
```

サブスクライバの追加、情報の出力、サブスクライバの削除

次に、サブスクライバ データベースにサブスクライバを追加し、その情報を取得して stdout に出力し、サブスクライバをサブスクライバ データベースから削除するプログラムを示します。

```
#include "SmApiBlocking.h"
#include <stdio.h>
int main(int argc, char* argv[])
{
checkArguments(argc,argv);
SmApiBlocking bapi;
//initiation
bapi.init();
bapi.setReplyTimeout(10000);   //set timeout for 10 seconds
bapi.connect(argv[1]);        // connect to the SM
//add subscriber
printf("adding subscriber to SM\n");
MappingType type = IP_RANGE;
char* customKey = "custom-key";
char* customVal = "10";
ReturnCode* ret = bapi.addSubscriber(
argv[2],   // name
&(argv[3]),// mapping
&type,     // mapping type
1,              // one mapping
&(argv[4]),// property key
&(argv[5]),// property value
1,              // number of properties
&customKey,//custom property key
&customVal,//custom property value
1,              // number of custom properties
argv[6]);  //domain
freeReturnCode (ret);
//Print subscriber
printf("Printing subscriber:\n");
ReturnCode* subfields = bapi.getSubscriber(argv[1]);
if (isReturnCodeError(subfields) == false)
{
printf("\tname:\t\t%s\n",
subfields->u.objectArray[0]->u.stringVal);
printf("\tmapping:\t%s\n",
subfields->u.objectArray[1]->u.stringArrayVal[0]);
printf("\tdomain:\t\t%s\n",
subfields->u.objectArray[3]->u.stringVal);
printf("\tautologout:\t%d\n",
subfields->u.objectArray[8]->u.intVal);
// Remove subscriber
printf("removing subscriber from SM\n");
bapi.removeSubscriber(argv[1]);
}
else
{
printf("error in subscriber retrieval\n");
}
freeReturnCode(subfields);
//finalization
bapi.disconnect();
return 0;
}
void checkArguments(int argc, char* argv[])
{
if (argc != 7)
{
printf("usage: AddPrintRemove <SM-address><subscriber-name>"
"<IP mapping><property-key><property-value><domain>");
exit(1);
}
}
```

ブロッキング API C コードの例

ここでは、次の 2 つのコード例を紹介します。

- サブスクライバ数の取得（p.3-39）
- サブスクライバの追加、情報の出力、サブスクライバの削除（p.3-40）

サブスクライバ数の取得

次の例では、SM データベース内のサブスクライバ総数と各サブスクライバ ドメイン内のサブスクライバ数を stdout に出力します。

```
include "SmApiBlocking_c.h"
include <stdio.h>
int main(int argc, char* argv[])
{
//initiation
SMB_HANDLE bapi = SMB_init(10,0,2000000,10,20);
if (bapi == NULL)
{
// init failure
return -1;
}
SMB_setReplyTimeout(bapi,300000); //set timeout for 5 minutes
SMB_connect(bapi,argv[1],14374); // connect to the SM
//operations
ReturnCode* domains = SMB_getDomains(bapi);
ReturnCode* totalSubscribers= SMB_getNumberOfSubscribers(bapi);
if ((isReturnCodeError(domains) == false) &&
(isReturnCodeError(totalSubscribers) == false))
{
printf("number of susbcribers in the database:\t\t %d\n",
totalSubscribers->u.intVal);
for (int i=0; i<domains->size; i++)
{
ReturnCode* numberOfSusbcribersInDomain=
SMB_getNumberOfSubscribersInDomain(bapi,
domains->u.stringArrayVal[i]);
if(isReturnCodeError(numberOfSusbcribersInDomain) == false
{
printf("number of susbcribers domain %s:\t\t%d\n",
domains->u.stringArrayVal[i],
numberOfSusbcribersInDomain->u.intVal);
}
freeReturnCode (numberOfSusbcribersInDomain);
}
}
freeReturnCode (domains);
freeReturnCode (totalSubscribers);
//finalization
SMB_disconnect(bapi);
SMB_release(bapi);
return 0;
}
```

サブスクライバの追加、情報の出力、サブスクライバの削除

次に、サブスクライバ データベースにサブスクライバを追加し、その情報を取得して **stdout** に出力し、サブスクライバをサブスクライバ データベースから削除するプログラムを示します。

```
include "SmApiBlocking_c.h"
include <stdio.h>
int main(int argc, char* argv[])
{
checkArguments(argc,argv);
//initiation
SMB_HANDLE bapi = SMB_init(10,0,2000000,10,20);
if (bapi == NULL)
{
// init failure
return -1;
}
SMB_setReplyTimeout(bapi,10000);  //set timeout for 10 seconds
SMB_connect(bapi,argv[1], 14374);// connect to the SM
//add subscriber
printf("adding subscriber to SM\n");
MappingType type = IP_RANGE;
char* customKey = "custom-key";
char* customVal = "10";
ReturnCode* ret = SMB_addSubscriber(
bapi,      // handle
argv[2],   // name
&(argv[3]), // mapping`
&type,     // mapping type
1,              // one mapping
&(argv[4]), // property key
&(argv[5]), // property value
1,              // number of properties
&customKey, //custom property key
&customVal, //custom property value
1,       // number of custom properties
argv[6]);   //domain
freeReturnCode (ret);
//Print subscriber
printf("Printing subscriber:\n");
ReturnCode* subfields = SMB_getSubscriber(bapi,argv[2]);
if (isReturnCodeError(subfields) == false)
{
printf("\tname:\t\t%s\n",
subfields->u.objectArray[0]->u.stringVal);
printf("\tmapping:\t%s\n",
subfields->u.objectArray[1]->u.stringArrayVal[0]);
printf("\tdomain:\t\t%s\n",
subfields->u.objectArray[3]->u.stringVal);
printf("\tautologout:\t%d\n",
subfields->u.objectArray[8]->u.intVal);
// Remove subscriber
printf("removing subscriber from SM\n");
SMB_removeSubscriber(bapi,argv[2]);
}
else
{
printf("error in subscriber retrieval\n");
}
freeReturnCode(subfields);
//finalization
SMB_disconnect(bapi);
SMB_release(bapi);
return 0;
}void checkArguments(int argc, char* argv[])
{
if (argc != 7)
{
printf("usage: AddPrintRemove <SM-address><subscriber-name>"
"<IP mapping><property-key><property-value><domain>");
exit(1);
}
}
```