CHAPTER 3

ブロッキング API

この章では、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説明します。

さらに、ブロッキング API のすべての操作を紹介し、コードの例も示します。

（注） *自動統合*の開発のみを行う場合は、この章を飛ばして、第 4 章「ノンブロッキング API」に進んでください。

マルチスレッドのサポート

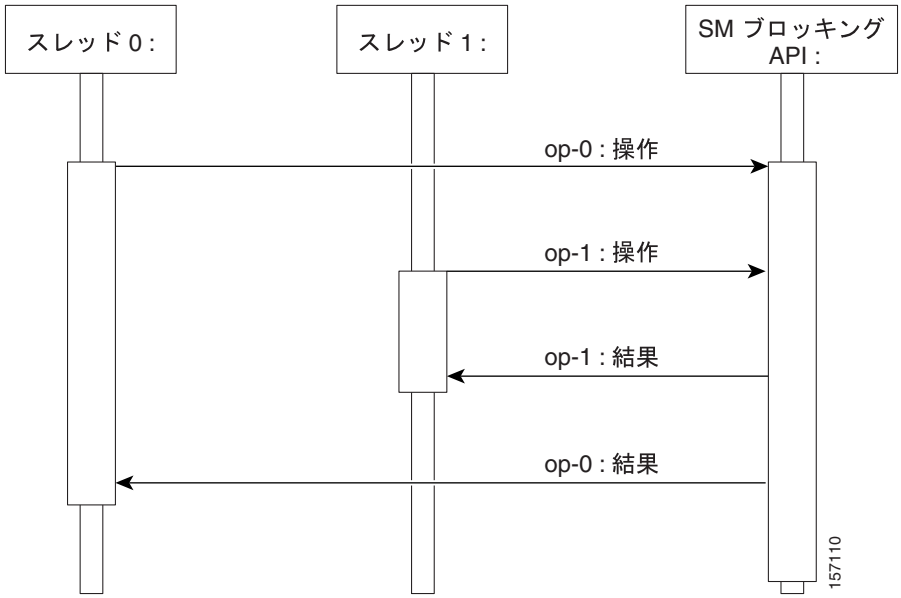
ブロッキング API では、メソッドを同時に呼び出すことのできるスレッド数に制限がありません。

（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**。

例

スレッド -0 が操作 -0 をタイム -0 で呼び出し、スレッド -1 が操作 -1 をタイム -1 で呼び出します。タイム -1 はタイム -0 よりあとです。この例では、次の図（縦が時間）のように操作 -1 が操作 -0 の**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 は各 API インターフェイスを処理するために 5 つのスレッドを割り当てます。5 つのスレッド順のスレッド番号のある API を使用するマルチスレッド アプリケーションを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、スレッドの呼び出し遅延が長くなる場合があります。

応答タイムアウトと操作タイムアウトの例外

ブロッキング操作が戻るのは、SM から操作結果が取得されたときだけです。ネットワークの異常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。SM API には、こうした状況を避ける手段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（**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と、**com.pcube.management.framework.rpc.OperationTimeoutException** が発生します。

応答タイムアウトを設定するには、long 値を指定して **setReplyTimeout** メソッドを呼び出します。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。

結果が戻るまで待機状態になって、呼び出し側をブロックしているメソッド呼び出しを別の方法で解放することも可能です。呼び出しているスレッドから **interrupt** メソッドを呼び出します。呼び出し側に **java.lang.InterruptedException** が戻ります。

ブロッキング API メソッド

ここでは、ブロッキング API のメソッドについて順番に説明します。各メソッドの構文のあとに、その入力パラメータと戻り値を記載します。

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、同じ操作の機能と構文構造はどちらの API でも同じです。

メソッドはどれも、SM との接続が確立される前に呼び出されると、**java.lang.IllegalStateException** を生じます。

ブロッキング API 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。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。
  - login（p.3-5）
  - logoutByName（p.3-8）
  - logoutByNameFromDomain（p.3-9）
  - logoutByMapping（p.3-10）
  - loginCable（p.3-11）
  - logoutCable（p.3-13）
- **サブスクライバの静的 / 手動設定** — GUI を利用する場合は、次のメソッドを使用します。
  - addSubscriber（p.3-14）
  - removeSubscriber（p.3-16）
  - removeAllSubscribers（p.3-17）
  - setPropertiesToDefault（p.3-27）
  - removeCustomProperties（p.3-28）
- サブスクライバ認識モードで独立して実行される単純な読み取りのみの操作の場合は、次のメソッドを使用します。
  - getNumberOfSubscribers（p.3-18）
  - getNumberOfSubscribersInDomain（p.3-18）
  - getSubscriber（p.3-19）
  - subscriberExists（p.3-20）
  - subscriberLoggedIn（p.3-21）
  - getSubscriberNameByMapping（p.3-22）
  - getSubscriberNames（p.3-23）
  - getSubscriberNamesInDomain（p.3-24）
  - getSubscriberNamesWithPrefix（p.3-25）
  - getSubscriberNamesWithSuffix（p.3-26）
  - getDomains（p.3-27）

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、メソッドの分類のみを目的としたものです。

- login（p.3-5）
- logoutByName（p.3-8）
- logoutByNameFromDomain（p.3-9）
- logoutByMapping（p.3-10）

login

構文

```
public void login(String subscriberName,
String[] mappings,
short[] mappingTypes,
String[] propertyKeys,
String[] propertyValues,
String domain,
boolean isMappingAdditive,
int autoLogoutTime)
throws InterruptedException, OperationTimeoutException, RpcErrorException
```

説明

The **login** メソッドは、SM データベース内にすでに存在しているサブスクライバのドメインおよびマッピング（場合によってはプロパティ）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与えたり、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

subscriberName — 「サブスクライバ名のフォーマット」（p.2-4）の説明を参照してください。

mappings — 「ネットワーク ID マッピング」（p.2-5）のマッピングおよびマッピング型に関する説明を参照してください。マッピングを指定せず、**isMappingAdditive** フラグが TRUE の場合、前のマッピングが保持されます。そのようなマッピングがない場合、操作はエラーとなります。

mappingTypes — 「ネットワーク ID マッピング」（p.2-5）のマッピングおよびマッピング型に関する説明を参照してください。

propertyKeys — 「サブスクライバ プロパティ」（p.2-7）のプロパティのキーおよび値に関する説明を参照してください。

propertyValues — 「サブスクライバ プロパティ」（p.2-7）のプロパティのキーおよび値に関する説明を参照してください。

domain — 「サブスクライバ ドメイン」（p.2-6）の説明を参照してください。

domain 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、サブスクライバに事前に割り当てられたドメインと異なる場合、サブスクライバは事前に割り当てられたドメインの SCE から自動的に削除され、新しいドメインの SCE へ移動されます。

`isMappingAdditive`

- **TRUE** — この呼び出しに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** — この呼び出しに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。

autoLogoutTime — 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のログアウトを実行します（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。

RPC 例外エラー コード

このメソッドで戻る可能性のあるエラー 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ドメイン　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 コードの説明については、「エラー コードのリスト」（p.A-1）を参照してください。

戻り値

なし

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、既存のマッピングを変更しない場合は、次のようにします。

```
login(
"john",                          // subscriber name
new String[]{"192.168.12.5"},
SMApiConstants.ALL_IP_MAPPINGS,
null, null,
"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、今までのマッピングを上書きするには、次のようにします。

```
login(
"john",                          // subscriber name
new String[]{"192.168.12.5"},
SMApiConstants.ALL_IP_MAPPINGS,
null, null,
"subscribers",                   // domain
false,                           // isMappingAdditive is false
-1);                             // autoLogoutTime set to infinite
```

以前 *john* へ割り当てられた 192.168.12.5 の自動ログアウト時間を延長するには、次にようにします。

```
login(
"john",                          //the previously assigned IP
new String[]{"192.168.12.5"},
SMApiConstants.ALL_IP_MAPPINGS,
null, null,
"subscribers",                   // domain
false,                           // isMappingAdditive
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。

```
login(
"john",
null, null,
new String[]{"packageId"}, // property key
new String[]{"10"},     // property value
"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）を変更するには、次のようにします。

```
login(
"john",
new String[]{"192.168.12.5"},
SMApiConstants.ALL_IP_MAPPINGS,
new String[]{"packageId"}, // property key
new String[]{"10"}, // property value
"subscribers", // domain
true,      // isMappingAdditive is set to true
-1);
```

logoutByName

構文

```
public boolean logoutByName(String subscriberName,
String[] mappings,
short[] mappingTypes)
throws InterruptedException, OperationTimeoutException, RpcErrorException
```

説明

データベース内でサブスクライバを検索し、マッピングを削除します。指定したサブスクライバが存在しなければ、何も実行されません。

パラメータ

subscriberName — 「サブスクライバ名のフォーマット」（p.2-4）の説明を参照してください。

mappings — 「ネットワーク ID マッピング」（p.2-5）のマッピングおよびマッピング型に関する説明を参照してください。マッピングを指定しないと、そのサブスクライバのすべてのマッピングが削除されます。

mappingTypes — 「ネットワーク ID マッピング」（p.2-5）のマッピングおよびマッピング型に関する説明を参照してください。

戻り値

- 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 データベースでそのサブスクライバが見つからなかった場合

RPC 例外エラー コード

このメソッドで戻る可能性のあるエラー 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 コードの説明については、「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* 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。

```
boolean isExist = logoutByName(
"john",
new String[]{"192.168.12.5"},
SMApiConstants.ALL_IP_MAPPINGS);
```

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

```
boolean isExist = logoutByName("john", null, null);
```

logoutByNameFromDomain

構文

```
public boolean logoutByNameFromDomain(String subscriberName,
String[] mappings,
short[] mappingTypes,
String domain)
throws InterruptedException, OperationTimeoutException, RpcErrorException
```

説明

logoutByName と似ていますが、呼び出し側はサブスクライバが所属するドメイン名も指定できます。サブスクライバのドメインがわかっている場合は、このメソッドを使うとパフォーマンスが向上します。

パラメータ

subscriberName — 「サブスクライバ名のフォーマット」（p.2-4）の説明を参照してください。

mappings — 「ネットワーク ID マッピング」（p.2-5）のマッピングおよびマッピング型に関する説明を参照してください。マッピングを指定しないと、そのサブスクライバのすべてのマッピングが削除されます。

mappingTypes — 「ネットワーク ID マッピング」（p.2-5）のマッピングおよびマッピング型に関する説明を参照してください。

domain — 「サブスクライバ ドメイン」（p.2-6）の説明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 NULL であるが、そのサブスクライバは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

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

RPC 例外エラー コード

このメソッドで戻る可能性のあるエラー 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 コードの説明については、「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。

```
boolean isExist = logoutByNameFromDomain(
"john",
new String[]{"192.168.12.5"},
SMApiConstants.ALL_IP_MAPPINGS,
"subscribers");
boolean isExist = logoutByNameFromDomain(
"john",
null,
null,
"subscribers");
```

logoutByMapping

構文

```
public boolean logoutByMapping(String mapping,
short mappingType,
String domain)
throws InterruptedException, OperationTimeoutException, RpcErrorException
```

説明

ドメインとマッピングに基づいてサブスクライバを探し、サブスクライバがデータベース内にあれば、そのマッピングを削除します。

パラメータ

mappings — 「ネットワーク ID マッピング」（p.2-5）のマッピングおよびマッピング型に関する説明を参照してください。

mappingTypes — 「ネットワーク ID マッピング」（p.2-5）のマッピングおよびマッピング型に関する説明を参照してください。

ドメイン — logoutByNameFromDomain 操作については、「パラメータ」（p.3-9）を参照してください。

戻り値

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

RPC 例外エラー コード

このメソッドで戻る可能性のあるエラー 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 コードの説明については、「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。

```
boolean isExist = logoutByMapping(
"192.168.12.5",
SMApiConstants. MAPPING_TYPE_IP,
"subscribers");
```

loginCable

構文

```
public void loginCable(String CPE,
String CM,
String IP,
int lease,
String domain,
String[] propertyKeys,
String[] propertyValues)
throws InterruptedException, OperationTimeoutException, RpcErrorException
```

説明

ケーブル環境に適応した login メソッド（SM 内のケーブル対応モジュールへの呼び出し）。このメソッドは CPE および CM の SM へのログイン用に設計されています。CPE ログインの場合は、CM 引数に CM MAC を、CPE 引数に CPE MAC を指定します。CM のログインでは、CPE と CM の両方の引数に CM MAC アドレスを指定します。CM が SM データベースに存在していない CPE のログインは無視されます。インポートまたは CM login 操作のいずれかによって、あらかじめ CM がデータベース内に存在している必要があります。詳細については、『*Cisco SCMS Subscriber Manager User Guide* 』の「CPE as Subscriber in Cable Environment」の章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（_）で連結したものとなります。呼び出し側で CPE と CM の値の長さが合計 **38** 文字を超えていないことを必ず確認してください。

パラメータ

CPE — CPE 固有の識別子（通常、MAC アドレス）

CM — ケーブル モデム固有の識別子（通常、MAC アドレス）

IP — CPE の IP アドレス

lease — CPE のリース時間

domain — 「サブスクライバ ドメイン」（p.2-6）の説明を参照してください。ドメインは通常、CMTS IP になります。

（注） CMT SIP 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。エイリアスの設定については、『*Cisco SCMS Subscriber Manager User Guide* 』の「Configuring Domains」の章を参照してください。

propertyKeys — 「サブスクライバ プロパティ」（p.2-7）のプロパティのキーおよび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 プロパティが部分的に指定されたり、まったく指定されない場合は、失われたプロパティの値は、その CPE が属している 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、そこに属す CPE のデフォルト値として使用されます。

propertyValues — 「サブスクライバ プロパティ」（p.2-7）のプロパティのキーおよび値に関する説明を参照してください。

戻り値

なし

RPC 例外エラー 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。

```
loginCable(
"CM1",
"CM1",
"192.168.12.5",
7200,          // lease time in seconds
"subscribers", null, null);
```

CM1 にある *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。

```
loginCable(
"CPE1",
"CM1",
"192.168.12.50",
3600,          // lease time in seconds
"subscribers", null, null);
```

logoutCable

構文

```
public boolean logoutCable(String CPE,
String CM,
String IP,
String domain)
```

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

CPE — 説明については、loginCable メソッドの項「パラメータ」（p.3-12）を参照してください。

CM — 説明については、loginCable メソッドの項「パラメータ」（p.3-12）を参照してください。

IP — 説明については、loginCable メソッドの項「パラメータ」（p.3-12）を参照してください。

domain — 説明については、loginCable メソッドの項「パラメータ」（p.3-12）を参照してください。

戻り値

- 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

RPC 例外エラー コード

なし

例

CM1 に属す *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。

```
boolean isExist = logoutCable(
"CPE1",
"CM1",
"192.168.12.5",
"subscribers");
```

addSubscriber

構文

```
public void addSubscriber(String subscriberName,
String[] mappings,
short[] mappingTypes,
String[] propertyKeys,
String[] propertyValues,
String[] customPropertyKeys,
String[] customPropertyValues,
String domain)
throws InterruptedException, OperationTimeoutException, RpcErrorException
```

説明

メソッド パラメータに準じて新規サブスクライバのレコードを作成し、SM データベースにレコードを追加します。

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** では指定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、渡されたパラメータによって、指定どおりにサブスクライバが設定されます。

（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します。

（注）　**addSubscriber** では、auto-logout 機能は常に無効になります。auto-logout を有効にするには、**login** を使用します。

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* を保持します。

AB の **addSubscriber** 操作がマッピングの指定なしに呼び出される場合（**mappings** および **mappingTypes** フィールドのいずれでも NULL）、*AB* はマッピングされません。

ただし、これらの NULL 値パラメータの **login** 操作を呼び出しても、*AB* のマッピングは変更しません。*AB* は *IP1* の前の IP マッピングを保持します。

パラメータ

subscriberName — 「サブスクライバ名のフォーマット」（p.2-4）の説明を参照してください。

mappings — 「ネットワーク ID マッピング」（p.2-5）のマッピングおよびマッピング型に関する説明を参照してください。

mappingTypes — 「ネットワーク ID マッピング」（p.2-5）のマッピングおよびマッピング型に関する説明を参照してください。

propertyKeys — 「サブスクライバ プロパティ」（p.2-7）のプロパティのキーおよび値に関する説明を参照してください。

propertyValues — 「サブスクライバ プロパティ」（p.2-7）のプロパティのキーおよび値に関する説明を参照してください。

customPropertyKeys — 「カスタム プロパティ」（p.2-7）のプロパティのキーおよび値に関する説明を参照してください。

customPropertyValues — 「カスタム プロパティ」（p.2-7）のプロパティのキーおよび値に関する説明を参照してください。

domain — 「サブスクライバ ドメイン」（p.2-6）の説明を参照してください。

NULL 値はサブスクライバにドメインがないことを示します。

domain が NULL であるが、そのサブスクライバには既にドメインがある場合、既存のドメインが保持されます。

戻り値

なし

RPC 例外エラー コード

このメソッドで返される可能性のあるエラー 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_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ています。
- ERROR_CODE_DATABASE_EXCEPTION

エラー コードの説明については、「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 プロパティをいくつか持つ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。

```
addSubscriber("john",
null, null,    // dynamic mappings will be set by login
null, null      // dynamic properties will be set by login
new String[]{  // custom property keys
        "work phone",
        "home phone"},
new String[]{  // custom property values
        "6543212"
        "5059927"},
"subscribers");// default domain
```

removeSubscriber

構文

```
public boolean removeSubscriber(String subscriberName)
throws InterruptedException, OperationTimeoutException, RpcErrorException
```

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

subscriberName — 「サブスクライバ名のフォーマット」（p.2-4）の説明を参照してください。

戻り値

- TRUE — データベースでそのサブスクライバが見つかり、正常に削除された場合
- FALSE — TRUE の条件が満たされなかった場合。つまり、データベースでその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

RPC 例外エラー コード

このメソッドで返される可能性のあるエラー 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 コードの説明については、「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。

```
boolean isExist = removeSubscriber("john");
```

removeAllSubscribers

構文

```
public void removeAllSubscribers()
throws InterruptedException, OperationTimeoutException, RpcErrorException
```

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があります。操作タイムアウトの例外条件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

なし

RPC 例外エラー コード

なし

getNumberOfSubscribers

構文

```
public int getNumberOfSubscribers()
throws InterruptedException, OperationTimeoutException, RpcErrorException
```

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

RPC 例外エラー コード

なし

getNumberOfSubscribersInDomain

構文

```
public int getNumberOfSubscribersInDomain(String domain)
throws InterruptedException, OperationTimeoutException, RpcErrorException
```

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

domain — 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内のサブスクライバ数

RPC 例外エラー コード

このメソッドで戻る可能性のあるエラー 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 コードの説明については、「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

```
public Object[] getSubscriber(String subscriberName)
throws InterruptedException, OperationTimeoutException, RpcErrorException
```

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、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」を参照）。

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、例外が発生します。

パラメータ

subscriberName — 「サブスクライバ名のフォーマット」（p.2-4）の説明を参照してください。

戻り値

9 つの要素を持つオブジェクト配列。次の表にインデックス値を示します。NULL 値となる配列要素はありません。

インデックス 0	サバスクライバ名（**java.lang.String**）
インデックス 1	マッピングの配列（**java.lang.String[]**）
インデックス 2	マッピング型の配列（**short[]**）

インデックス 3	ドメイン名（**java.lang.String**）
インデックス 4	プロパティ名の配列（**java.lang.String[]**）
インデックス 5	プロパティ値の配列（**java.lang.String[]**）
インデックス 6	カスタム プロパティ名の配列（**java.lang.String[]**）
インデックス 7	カスタム プロパティ値の配列（**java.lang.String[]**）
インデックス 8	その時点からの秒数で表される自動ログアウト時間 （設定されていない場合は -1）（**long[]**）

RPC 例外エラー コード

このメソッドで戻る可能性のあるエラー コードは次のとおりです。

- ERROR_CODE_SUBSCRIBER_DOES_NOT_EXIST
- ERROR_CODE_DATABASE_EXCEPTION

エラー コードの説明については、「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。

```
Object[] subRecord = getSubscriber("john");
String[] mappings = (String[])subRecord[1]
short[] mappingTypes = {short[])subRecord[2];
String domainName = (String)subRecord[3];
String[] propertyNames = (String[])subRecord[4];
String[] propertyValues = (String[])subRecord[5];
String[] customPropertyName = (String[])subRecord[6];
String[] customPropertyValues = (String[])subRecord[7];
long[] autoLogoutTime = (long[])subRecord[8];
```

subscriberExists

構文

```
public boolean subscriberExists(String subscriberName)
throws InterruptedException, OperationTimeoutException, RpcErrorException
```

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

subscriberName — 「サブスクライバ名のフォーマット」（p.2-4）の説明を参照してください。

戻り値

- 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

RPC 例外エラー コード

なし

subscriberLoggedIn

構文

```
public boolean subscriberLoggedIn(String subscriberName)
throws InterruptedException, OperationTimeoutException, RpcErrorException
```

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 データベースに存在しているかどうかを確認します。

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**。確かなのは、必要に応じてそのサブスクライバに SCE による pull が実行可能であることだけです。

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、例外が発生します。

パラメータ

subscriberName — 「サブスクライバ名のフォーマット」（p.2-4）の説明を参照してください。

戻り値

- 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

RPC 例外エラー コード

このメソッドで戻る可能性のあるエラー コードは次のとおりです。

- ERROR_CODE_ILLEGAL_SUBSCRIBER_NAME
- ERROR_CODE_DATABASE_EXCEPTION

エラー コードの説明については、「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

```
public String getSubscriberNameByMapping(String mapping,
short mappingType,
String domain)
throws InterruptedException, OperationTimeoutException, RpcErrorException
```

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を検索します。

パラメータ

mappings — 「ネットワーク ID マッピング」（p.2-5）のマッピングおよびマッピング型に関する説明を参照してください。

mappingTypes — 「ネットワーク ID マッピング」（p.2-5）のマッピングおよびマッピング型に関する説明を参照してください。

domain — 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 NULL であるが、そのサブスクライバは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

- 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 — サブスクライバ レコードが見つからなかった場合

RPC 例外エラー コード

このメソッドで戻る可能性のあるエラー 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 コードの説明については、「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

```
public String[] getSubscriberNames(String lastBulkEnd,
int numOfSubscribers)
throws InterruptedException, OperationTimeoutException, RpcErrorException
```

説明

SM データベースから、ある 1 まとまり（バルク）のサブスクライバ名を取得します。**lastBulkEnd** から始まり、アルファベット順に **numOfSubscribers** だけサブスクライバが取得されます。

lastBulkEnd 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（全バルクの）サブスクライバの合計数であるとは**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

lastBulkEnd — 最後のバルクの最後のサブスクライバ名。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。

numOfSubscribers — 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**推奨できません**。

戻り値

アルファベット順のサブスクライバ名の配列

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、**numOfSubscribers** と SM 制限値（1000）のうち小さい方の値に制限されます。

RPC 例外エラー コード

このメソッドで返される可能性のあるエラー コードは次のとおりです。

- ERROR_CODE_ILLEGAL_SUBSCRIBER_NAME
- ERROR_CODE_DATABASE_EXCEPTION

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

例

```
boolean hasMoreSubscribers;
String lastBulkEnd = null;
int bulkSize = 100;
do {
String[] subscribers = smApi.getSubscriberNames(lastBulkEnd, bulkSize);
hasMoreSubscribers = false;
if (subscribers != null) {
for (int i = 0; i <subscribers.length; i++) {
// do something with subscribers[i]
}
if (subscribers.length == bulkSize) {
hasMoreSubscribers = true;
lastBulkEnd = subscribers[bulkSize - 1];
}
}
} while (hasMoreSubscribers);
```

getSubscriberNamesInDomain

構文

```
public String[] getSubscriberNamesInDomain(String lastBulkEnd,
int numOfSubscribers,
String domain)
throws InterruptedException, OperationTimeoutException, RpcErrorException
```

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。

この操作の意味は、「getSubscriberNames」（p.3-23）操作の意味と同じです。

パラメータ

lastBulkEnd — getSusbcriberNames 操作の「パラメータ」（p.3-23）の項を参照してください。

numOfSubscribers — getSusbcriberNames 操作の「パラメータ」（p.3-23）の項を参照してください。

domain — 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

getSubscriberNames 操作の「戻り値」（p.3-23）の項を参照してください。

RPC 例外エラー コード

このメソッドで戻る可能性のあるエラー 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 コードの説明については、「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

```
public String[] getSubscriberNamesWithPrefix(String lastBulkEnd,
int numOfSubscribers,
String prefix)
throws InterruptedException, OperationTimeoutException, RpcErrorException
```

説明

指定されたプレフィッ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。

この操作の意味は、「getSubscriberNames」（p.3-23）操作の意味と同じです。

パラメータ

lastBulkEnd — getSusbcriberNames 操作の「パラメータ」（p.3-23）の項を参照してください。

numOfSubscribers — getSusbcriberNames 操作の「パラメータ」（p.3-23）の項を参照してください。

prefix — 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（大文字と小文字は区別されます）

戻り値

要求されたプレフィッ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

getSubscriberNames 操作の「戻り値」（p.3-23）の項を参照してください。

RPC 例外エラー コード

このメソッドで戻る可能性のあるエラー コードは次のとおりです。

- ERROR_CODE_ILLEGAL_SUBSCRIBER_NAME
- ERROR_CODE_DATABASE_EXCEPTION

エラー コードの説明については、「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

```
public String[] getSubscriberNamesWithSuffix(String lastBulkEnd,
int numOfSubscribers,
String suffix)
throws InterruptedException, OperationTimeoutException, RpcErrorException
```

説明

指定されたサフィックスで名前が終わるサブスクライバを SM データベースから取得します。

この操作の意味は、「getSubscriberNames」（p.3-23）操作の意味と同じです。

パラメータ

lastBulkEnd — getSusbcriberNames 操作の「パラメータ」（p.3-23）の項を参照してください。

numOfSubscribers — getSusbcriberNames 操作の「パラメータ」（p.3-23）の項を参照してください。

suffix — 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（大文字と小文字は区別されます）

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

getSubscriberNames 操作の「戻り値」（p.3-23）の項を参照してください。

RPC 例外エラー コード

このメソッドで戻る可能性のあるエラー コードは次のとおりです。

- ERROR_CODE_ILLEGAL_SUBSCRIBER_NAME
- ERROR_CODE_DATABASE_EXCEPTION

エラー コードの説明については、「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

```
public String[] getDomains()
throws InterruptedException, OperationTimeoutException, RpcErrorException
```

説明

SM ドメイン リポジトリ内の最新のサブスクライバ 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 ドメイン名の完全なリスト

RPC 例外エラー コード

なし

setPropertiesToDefault

構文

```
public void setPropertiesToDefault(String subscriberName,
String[] properties)
throws InterruptedException, OperationTimeoutException, RpcErrorException
```

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、**java.lang.IllegalStateException** が戻ります。

パラメータ

subscriberName — 「サブスクライバ名のフォーマット」（p.2-4）の説明を参照してください。

properties — 「サブスクライバ プロパティ」（p.2-7）のプロパティのキーおよび値に関する説明を参照してください。

戻り値

なし

RPC 例外エラー コード

このメソッドで戻る可能性のあるエラー 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 コードの説明については、「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

```
public void removeCustomProperties(String subscriberName,
String[] customProperties)
throws InterruptedException, OperationTimeoutException, RpcErrorException
```

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

subscriberName — 「サブスクライバ名のフォーマット」（p.2-4）の説明を参照してください。

CustomProperties — 「カスタム プロパティ」（p.2-7）のプロパティのキーおよび値に関する説明を参照してください。

戻り値

なし

RPC 例外エラー コード

このメソッドで戻る可能性のあるエラー 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 コードの説明については、「エラー コードのリスト」（p.A-1）を参照してください。

ブロッキング API のコード例

ここでは、次に示す 2 つのコード例を紹介します。

- サブスクライバ数の取得
- サブスクライバの追加、サブスクライバ情報の出力、サブスクライバの削除
- サブスクライバ数の取得（p.3-30）
- 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.3-31）

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。

```
package blocking;
import com.pcube.management.api.SMBlockingApi;
public class PrintInfo {
public static void main (String args[]) throws Exception {
SMBlockingApi bapi = new SMBlockingApi();
try {
//initiation
bapi.setReplyTimeout(300000);  //set timeout for 5 minutes
bapi.connect(args[0]);         // connect to the SM
//operations
String[] domains=bapi.getDomains();
int totalSubscribers=bapi.getNumberOfSubscribers();
System.out.println(
"number of susbcribers in the database:\t\t "+
totalSubscribers);
for (int i=0; i<domains.length; i++) {
int numberOfSusbcribersInDomain=
bapi.getNumberOfSubscribersInDomain(domains[i]);
System.out.println(
"number of susbcribers domain "+domains[i]+
":\t\t "+numberOfSusbcribersInDomain);
}
} finally {
//finalization
bapi.disconnect();
}
}
}
```

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

次に示すのは、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、最後にその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です。

```
package blocking;
import com.pcube.management.api.SMBlockingApi;
import com.pcube.management.api.SMApiConstants;
public class AddPrintRemove {
public static void main (String args[]) throws Exception {
checkArguments(args);
SMBlockingApi bapi = new SMBlockingApi();
try {
//initiation
bapi.setReplyTimeout(10000);   //set timeout for 10 seconds
bapi.connect(args[0]);         // connect to the SM
//add subscriber
System.out.println("+ adding subscriber to SM");
bapi.addSubscriber(
args[1],                    //name
new String[]{args[2]},      //mapping`
SMApiConstants.ALL_IP_MAPPINGS,
new String[]{args[3]},      //property key
new String[]{args[4]},      //property value
new String[]{"custom-key"}, //custom property key
new String[]{"10"},         //custom property value
args[5]);                   //domain
//Print subscriber
System.out.println("+ Printing subscriber");
Object[] subfields = bapi.getSubscriber(args[1]);
System.out.println("\tname:\t\t"+subfields[0]);
System.out.println("\tmapping:\t"+
((String[])subfields[1])[0]);
System.out.println("\tdomain:\t\t"+subfields[3]);
System.out.println("\tautologout:\t"+subfields[8]);
//Remove subscriber
System.out.println("+ removing subscriber from SM");
bapi.removeSubscriber(args[1]);
} finally {
//finalization
bapi.disconnect();
}
}
static void checkArguments(String[] args)  throws Exception{
if (args.length != 6) {
System.err.println(
"usage: java AddPrintRemove <SM-address>"+
" <subscriber-name><IP mapping><property-key>"+
" <property-value><domain>");
System.exit(1);
}
}
}
```