CHAPTER 22

IMGW Device Module Development Toolkit

Intelligent Modular Gateway（IMGW）Device Module Development Toolkit では、IMGW のサウスバウンド インターフェイスが明確に定義されています。また、デバイス モジュールの Cisco Configuration Engine への実装後、プラグイン デバイス モジュールを IMGW に登録するための登録ユーティリティが提供されています。

ここでは、IMGW Device Module Development Toolkit の要件を分析し、このツールキットによって提供される機能について説明します。

（注） デバイス モジュールは、シェル スクリプトまたは Linux や Solaris の実行ファイルでインストールすることもできます。ただし、そのデバイス モジュールは IMGW サウスバウンド インターフェイスに準拠している必要があります。

ユーザ タイプ

このツールキットは、次の 3 タイプのユーザを対象としています。

- *プラグイン開発者* : このツールキットで定義されている IMGW サウスバウンド インターフェイスに準拠するデバイス モジュールを開発する担当者
- *システム管理者* : 次の操作の担当者
  - Cisco Configuration Engine へのデバイス モジュールの組み込みと取り外し
  - プラグイン デバイス モジュールの登録と登録解除
  - Cisco Configuration Engine 上のデバイス モジュールの更新
- *ネットワーク オペレータ* : プラグイン デバイス モジュールを使用してデバイスを設定するオペレータ

ツールキットの使用方法

このツールキットには、次の 3 つの一般的な使用方法があります。

- デバイス モジュールを Cisco Configuration Engine に組み込み、そのデバイス モジュールを使用してデバイスを設定する。
- Cisco Configuration Engine 上のデバイス モジュールを更新し、変更後のデバイス モジュールを使用してデバイスを設定する。

- Cisco Configuration Engine からデバイス モジュールを取り外す。

Cisco Configuration Engine へのデバイス モジュールの組み込み

ステップ 1　*プラグイン開発者*は、所定のデバイス タイプを処理するために、このツールキットで定義されている IMGW サウスバウンド インターフェイスに準拠するデバイス モジュールを開発します。

デバイス モジュール構文の詳細については、「IMGW サウスバウンド インターフェイス」（P.22-2）を参照してください。

ステップ 2　*システム管理者*がデバイス モジュールを Cisco Configuration Engine に実装します。

ステップ 3　*システム管理者*が登録ユーティリティを実行してデバイス モジュールを IMGW に登録します。

ステップ 4　*ネットワーク オペレータ*がデバイス モジュールを使用してデバイスを設定します。

Cisco Configuration Engine 上のデバイス モジュールの更新

ステップ 1　*プラグイン開発者*が新しいバージョンのデバイス モジュールを提供します。

ステップ 2　*システム管理者*が登録ユーティリティを実行して IMGW からデバイス モジュールを登録解除します。

更新するデバイス モジュールが登録されていない場合は、この手順を省略します。

ステップ 3　*システム管理者*が Cisco Configuration Engine 上のデバイス モジュールを新しいバージョンに更新します。

ステップ 4　*システム管理者*が登録ユーティリティを実行して更新されたデバイス モジュールを IMGW に登録します。

ステップ 5　*ネットワーク オペレータ*が変更されたデバイス モジュールを使用してデバイスを設定します。

Cisco Configuration Engine からのデバイス モジュールの取り外し

ステップ 1　*システム管理者*が登録ユーティリティを実行して IMGW からプラグイン デバイス モジュールを登録解除します。

ステップ 2　*システム管理者*がプラグイン デバイス モジュールを Cisco Configuration Engine から取り外します。

IMGW サウスバウンド インターフェイス

コマンド実行または設定更新が行われたことが IMGW ランタイムに受信されると、デバイス情報データベースから最初にデバイス タイプ情報が取得されます。デバイス タイプおよびオペレーション タイプ（**CONFIG_UPLOAD** または **CONFIG_DOWNLOAD**）に対応するデバイス モジュールが登録されている場合、IMGW ランタイムはプロセスを分岐して適切なプラグイン プログラムを実行し、プラグイン プログラムにパラメータ リストを渡します。

<device type、operation type> のペアからプラグイン プログラムへの初期のマッピング情報は、起動時にコンフィギュレーション ファイルからメモリに読み込まれます。IMGW の実行時でも、システム管理者はツールキットの登録ユーティリティを使用して、マッピング情報のエントリを追加、削除、または更新することができます。

*システム管理者*は非レガシー デバイス モジュールのエントリだけを変更できます。この制限は、IMGW ランタイムで強制されます。

ユーザ設計のデバイス モジュールの仕様

ユーザ定義のデバイス モジュールは、ここで示すように IMGW サウスバウンド インターフェイスに準拠している必要があります。

設定イベント

*<**plug-in program**> <temp_logfile_name> <logging_level> <device_id> <action_type> <warning_logfile_name> <error_logfile_name> <hop_information_string> <configuration_file_name> <persistence> <operation_timeout_value> <prompt_timeout_value>*

実行イベント

*<**plug-in program**> <temp_logfile_name> <logging_level> <device_id> <action_type> <hop_information_string> <command_to_be_executed> <command_arguments> <exec_response_logfile_name> <operation_timeout_value> <prompt_timeout_value>*

ホップ テスト

*<**plug-in program**> <temp_logfile_name> <logging_level> <device_id> <action_type> <hop_information_string> <operation_timeout_value> <prompt_timeout_value>*

（注） IMGW サウスバウンド インターフェイスに指定されたすべてのファイルは、IMGW ランタイムによって管理され、ファイル名は絶対パス名です。

パラメータの説明

Plug-in Program : IMGW ランタイムによって分岐された子プロセスで実行されるプラグイン プログラム。システム管理者が登録時にこの情報を IMGW ランタイムに提供します。

temp_logfile_name : デバイス モジュールの一時ログ ファイルへのフル パス。デバイス モジュールはこのパスを使用して、オペレーションの 1 つのインスタンス（設定のダウンロード、コマンドの実行、ホップのテスト）の処理履歴をログに記録する必要があります。デフォルトでは、このファイルは Cisco Configuration Engine の */tmp* ディレクトリにあります。プラグイン プログラムの終了後、IMGW ランタイムはこのファイルの内容をデバッグに使用するために、*/opt/CSCOimgw/bin/IMGW-DEVMOD_LOG* という集中ログ ファイルに記録し、その後このファイルをリンク解除します。

logging_level : verbose、error、または silent を値として使用。このフラグは、ホスト システム上でセットアップ コマンドを実行することによって設定できます。指定したロギング レベルに基づいて、デバイス モジュールが情報を *<temp_logfile_name>* ファイルに記録するように設定することを推奨します。

device_id : デバイス モジュールによって処理されるデバイスの ID。この ID は、*cisco.mgmt.cns.config.load* または *cisco.mgmt.cns.exec.cmd* イベントによって渡されます。

action_type : config, **exec** または **hoptest**。アクション タイプ **config** を指定すると、デバイス モジュールはデバイス設定を更新するように通知されます。アクション タイプ **exec** を指定すると、デバイス モジュールはデバイス上でコマンドを実行するように通知されます。アクション タイプ **hoptest** を指定すると、デバイス モジュールは、<*hop_information_string*> に指定されたホップ情報を使用してデバイスにアクセスできるかどうかをテストするように通知されます。デバイス モジュールは、このフラグに応じて適切な操作を実行する必要があります。

warning_logfile_name : すべての警告メッセージおよびそれらに対応するコンフィギュレーション コマンドの行番号を記録するためにデバイス モジュールによって使用されるファイルへのフル パス。このパラメータは、アクション タイプが **config** の場合にだけ、IMGW ランタイムによって提供されます。このファイルの情報は、設定が警告付きで成功した場合に *cisco.mgmt.cns.config.load* イベントへの応答メッセージを生成する目的でだけ使用されるためです。IMGW ランタイムが適切な応答メッセージを生成するためには、各警告メッセージは新しい行で開始され、**LINE <** *警告メッセージの原因となったコンフィギュレーション コマンドの行番号*>**:** が先頭に付いている必要があります。警告ファイルの例を次に示します。

```
LINE 3: The interface has already been removed

.
.
.

LINE 7: The interface already exists.
```

このファイルの場所は、ホスト システムの */tmp* の下です。プラグイン プログラムが終了した後、IMGW ランタイムが応答イベント ペイロード内にこのファイルの内容を入力し、次にこのファイルをすぐにリンク解除します。

error_logfile_name : エラー メッセージの発生および対応するコンフィギュレーション コマンドの行番号を記録するためにデバイス モジュールによって使用されるファイルへのフル パス。このパラメータは、アクション タイプが **config** の場合にだけ、IMGW ランタイムによって提供されます。このファイルの情報は、設定が失敗した場合に *cisco.mgmt.cns.config.load* イベントへの応答メッセージを生成する目的でだけ使用されるためです。IMGW ランタイムが適切な応答メッセージを生成するためには、各警告メッセージは新しい行で開始され、**LINE** <*エラー メッセージの原因となったコンフィギュレーション コマンドの行番号*>: が先頭に付いている必要があります。

エラー ファイルの例を次に示します。

```
LINE 3: % Invalid input detected at

LINE 7: % Incomplete command

.
.
.

LINE 12: % The interface already exists
```

このファイルの場所は、ホスト システムの */tmp* の下です。プラグイン プログラムが終了した後、IMGW ランタイムが応答イベント ペイロード内にこのファイルの内容を入力し、次にこのファイルをすぐにリンク解除します。

exec_response_logfile_name : デバイスに対するコマンド実行の出力を記録するために使用されるファイルへのフル パス。このパラメータは、アクション タイプが **exec** の場合にだけ、IMGW ランタイムによって提供されます。このファイルの場所は、ホスト システムの */tmp* の下です。プラグイン プログラムが終了した後、IMGW ランタイムが応答イベント ペイロード内にこのファイルの内容を入力し、次にこのファイルをすぐにリンク解除します。

hop_information_string : デバイスのアクセス情報を保存するために使用される文字列。デバイスの個々のホップ情報をすべて、順番に並べた文字列連結です。ホップ情報とその *<hop_information_string>* の例を次に示します。

ホップ タイプ	IP アドレス	ポート	ユーザ名	パスワード
IOS_LOGIN	172.29.145.45		Admin	Cisco
IOS_EN			Lab	Lab

対応する *<hop_information_string>* は次のようになります。

```
"IOS_LOGIN" "172.29.145.45" " " "Admin" "Cisco" "IOS_EN" " " " " "Lab" "Lab"
```

（注） ヌル値を含むこれらのホップ情報フィールドには、子プロセスへ渡す前に、IMGW ランタイムが自動的にスペースを追加します。

command_to_be_executed : デバイスに対して実行されるコマンド。アクション タイプが **exec** の場合にのみ、IMGW ランタイムによって提供されます。

command_arguments : デバイスに対して実行されるコマンドの引数。アクション タイプが **exec** の場合にのみ、IMGW ランタイムによって提供されます。

configuration_file_name : デバイスにダウンロードされるコンフィギュレーション ファイルへのフルパス。このパラメータは、アクション タイプが **config** の場合にだけ、IMGW ランタイムによって提供されます。このファイルの場所は、ホスト システムの */tmp* の下です。プラグイン プログラムの終了後、IMGW ランタイムはただちにこのファイルをリンク解除します。

persistence : y または **n**。**y** を値として指定すると、設定が不揮発性ストレージに書き込まれる必要があることを意味します。アクション タイプが **config** の場合にのみ、IMGW ランタイムによって提供されます。このオプションは、デバイス タイプによって異なります。つまり、デバイス タイプがこのオプションをサポートしていない場合、デバイス モジュールはこのオプションを無視できます。

operation_timeout_value : デバイスに対してコマンドを実行できる最長時間。このパラメータは、現在 IOS、CatOS、CatIOS、PIX、CSS および CE デバイス用の IMGW レガシー デバイス モジュールにある Expect スクリプトで使用されています。ユーザ定義のデバイス モジュールでこのパラメータを使用しない場合は、このパラメータを無視できます。

prompt_timeout_value : デバイスへのログイン セッション中に、次のプロンプトまで待機できる最長時間。このパラメータは、現在 IOS、CatOS、CatIOS、PIX、CSS および CE デバイス用の IMGW レガシー デバイス モジュールにある Expect スクリプトで使用されています。ユーザ定義のデバイス モジュールでこのパラメータを使用しない場合は、このパラメータを無視できます。

終了コード

分岐されたプロセス（プラグイン プログラムが実行されるプロセス）が終了したとき、IMGW ランタイムは次の終了コードを分岐されたプロセスから返します。

設定イベント：

0 : ダウンロードの成功

1 : ダウンロードの失敗

2 : ダウンロードが警告メッセージ付きで成功

実行イベント：

0：コマンド実行の成功

1：コマンド実行の失敗

ホップ テスト：

0：ホップ テストの成功

1：ホップ テストの失敗

プラグイン デバイス モジュールの開発方法

このツールキットを使用することにより、*プラグイン開発者*は、デバイス モジュールが「IMGW サウスバウンド インターフェイス」（P.22-2）で示されている IMGW サウスバウンド インターフェイスに準拠している限り、任意の実装方法でプラグイン デバイス モジュールを実現できます。

このツールキットでは、Perl スクリプトおよび Expect スクリプトのサンプル コード（「ツールキットの使用方法」（P.22-1）を参照）やインライン コメントも提供されており、初心者がプラグイン デバイス モジュールのワークフローを理解するのに役立ちます。

プラグイン デバイス モジュールは、次の 3 つの基本機能を提供します。

- デバイス設定の更新
- コマンドの実行
- ホップ テスト

最初の 2 つの機能はそれぞれ *cisco.mgmt.cns.config.load* イベントおよび *cisco.mgmt.cns.exec.cmd* イベントへの応答です。最後の機能は、IMGW ランタイムが必要とする内部のルーチン処理のため、ネットワーク オペレータの関与を必要としません。

IMGW ランタイムが子プロセスを発生してプラグイン プログラムを実行した後、対応するデバイス モジュールがパラメータ リストからアクション タイプを読み込む必要があります。アクション タイプによる条件を次に示します。

- **config**：デバイス モジュールはデバイス設定を更新する必要があります。
- **exec**：デバイス モジュールはコマンドの実行を行う必要があります。
- **hoptest**：デバイス モジュールはホップ テストを行う必要があります。

開発に関するガイドライン

次の各項目では、各機能と関連付けられているプロセスについて説明します。

（注）　次の各項目で示すアクションの主体は、プラグイン デバイス モジュールです。

デバイス設定の更新

1. <*hop_information_string*> を使用してデバイスにアクセスします。
2. <*configuration_file_name* > で指定したコンフィギュレーション ファイルをデバイス上にダウンロードします。
3. 上記のダウンロード操作に成功した場合、<*persistence*> は **y** に設定され、デバイスはこのオプションをサポートし、非揮発性ストレージに設定を書き込みます。

4. デバイスによってプロンプトされたすべての警告メッセージおよび対応するコンフィギュレーション コマンドの行番号を、<*warning_logfile_name*> で指定したファイルに指定形式で書き込みます（「パラメータの説明」（P.22-3）を参照）。このファイルの内容は、ダウンロードが警告メッセージ付きで成功した場合、応答イベントのペイロードの一部になります。
5. デバイスによってプロンプトされたすべてのエラー メッセージおよび対応するコンフィギュレーション コマンドの行番号を、<*error_logfile_name*> で指定したファイルに指定形式で書き込みます（「パラメータの説明」（P.22-3）を参照）。最初のエラー メッセージおよび対応するコンフィギュレーション コマンドの行番号は、ダウンロードに失敗した場合、応答イベントのペイロードの一部になります。
6. <*logging_level*> に基づき、プロシージャ全体を通してデバッグするために、<*temp_logfile_name*> で指定したファイルに対して処理履歴を選択的にリダイレクトします。
7. 適切な終了コードを使用して終了し、IMGW ランタイムに制御を返します。終了コードの定義の取得方法については、「終了コード」（P.22-5）を参照してください。

コマンドの実行

1. <*hop_information_string*> を使用してデバイスにアクセスします。
2. <*command_to_be_executed*> に <*command_arguments*> を指定し、デバイスに対して実行します。
3. コマンド実行からのすべての出力を <*exec_response_logfile_name*> に指定したファイルに書き込みます。このファイルの内容は、応答イベントのペイロードの一部になります。
4. <*logging_level*> に基づき、プロシージャ全体を通してデバッグするために、<*temp_logfile_name*> で指定したファイルに対して処理履歴を選択的にリダイレクトします。
5. 適切な終了コードを使用して終了し、IMGW ランタイムに制御を返します。終了コードの定義の取得方法については、「終了コード」（P.22-5）を参照してください。

ホップ テスト

1. <*hop_information_string*> を使用してデバイスにアクセスします。
2. <*logging_level*> に基づき、プロシージャ全体を通してデバッグするために、<*temp_logfile_name*> で指定したファイルに対して処理履歴を選択的にリダイレクトします。
3. 適切な終了コードを使用して終了し、IMGW ランタイムに制御を返します。終了コードの定義の取得方法については、「終了コード」（P.22-5）を参照してください。

プラグイン デバイス モジュールのインストール

*システム管理者*は、インストールおよびアンインストールを担当する必要があります。登録ユーティリティをコールする前に、インストールが成功していることを確認してください。

システム管理者は、すべてのプラグイン デバイス モジュールを予約ファイル ディレクトリである */opt/ConfigEngine/CSCOimgw/plugin-modules* にインストールします。このとき、デバイス モジュールごとに 1 つのサブディレクトリを使用します。たとえば、MGX 用のデバイス モジュールを */opt/ConfigEngine/CSCOimgw/plugin-modules/MGX* にインストールし、NT 用のデバイス モジュールを */opt/ConfigEngine/CSCOimgw/plugin-modules/NT* にインストールします。

*システム管理者*は、デバイス モジュール インストール ディレクトリ内のみを操作してモジュールの実行環境を設定および解除する必要があります。インストール アクティビティが Cisco Configuration Engine 上の他のコンポーネントの実行環境に影響を与えないようにするためです。

プラグイン デバイス モジュールの登録

*システム管理者*は、デバイス モジュールの登録時にデバイス タイプとプラグイン プログラムへのフルパスを入力する必要があります。IMGW ランタイムはこの情報の整合性をチェックしません。情報が正しいことを確認するのは、*システム管理者*の責任です。

このツールキットはシステム管理者に動的な登録ユーティリティを提供します。これを使用すると、*システム管理者*は、IMGW ランタイムを壊すことなくシームレスにデバイス モジュールを IMGW に組み込みおよび取り外しできます。したがって、登録および登録解除するデバイス モジュールと関連のないサービスは影響を受けません。ただし、それ以外のサービスに関しては、これが当てはまらない場合があります。

たとえば、デバイス モジュール *x* に対して登録解除コマンドを発行すると、*x* と関連する、イベントバスのキューに入ったままのイベントに、IMGW から失敗応答が返される場合があります。

注意　*システム管理者*は、次に実行する登録アクティビティをあらかじめすべての*ネットワーク オペレータ*に通知することを強く推奨します。そうすることで、*ネットワーク オペレータ*は関連操作を事前に停止できるようになります。

エンド ユーザ インターフェイス

IMGW Device Module Development Toolkit のエンド ユーザ インターフェイスは、IMGW サウスバウンド インターフェイスとコマンドライン登録ユーティリティで構成されています。

設定と制約事項

このツールキットでは、IMGW に追加できるプラグイン デバイス モジュールの最大数に制限はありません。

デバイス モジュールの制約事項

- デバイス モジュールは、Linux プラットフォームまたは Solaris プラットフォーム（あるいはその両方）で実行できます。
- デバイス モジュールの実行ファイルが C++ バイナリ ファイルの場合、Cisco Configuration Engine にある glib を適用可能な箇所で使用する必要があります。
- デバイス モジュールの実行ファイルが Java クラスの場合、Cisco Configuration Engine の既存 JVM 内で実行する必要があります。
- デバイス モジュールに Perl スクリプトまたは Expect スクリプト（あるいはその両方）が含まれている場合、そのスクリプトには、Cisco Configuration Engine に存在する Perl インタープリタまたは Expect インタープリタ（あるいはその両方）を使用する必要があります。

登録ユーティリティの制約事項

*システム管理者*は、IMGW レガシー デバイス モジュールを登録または登録解除しないでください。ユーザ固有のニーズを満たすために、レガシー デバイス モジュールのいずれかを変更して、CatOS、CatIOS、PIX、CSS、CE、または IOS の各デバイスでアップロードおよびダウンロード操作を実行することが必要になる場合もあります。この場合は、レガシー デバイス モジュールのユーザ自身が所有するコピーを変更し、別のデバイス タイプ名を変更後のデバイス モジュールに関連付けてから、そのデバイス モジュールを IMGW に登録します。