



## CHAPTER 3

# SIP メッセージ API

---

Lua スクリプト作成環境には、メッセージの処理を可能にする一連の API が用意されています。これらの API について、次のカテゴリで説明します。

- 「要求行または応答行での操作」(P.3-1)
- 「ヘッダーの操作」(P.3-4)
- 「SDP の操作」(P.3-14)
- 「コンテンツ本文の操作」(P.3-17)
- 「メッセージのブロック」(P.3-19)
- 「透過性」(P.3-20)
- 「Per-Dialog コンテキストの管理」(P.3-20)
- 「ユーティリティ」(P.3-22)

## 要求行または応答行での操作

- `getRequestLine`
- `getRequestUriParameter`
- `setRequestUri`
- `getResponseLine`
- `setResponseCode`

### `getRequestLine`

`getRequestLine()` returns the method, request-uri, and version

このメソッドは次の 3 つの値を返します。

- メソッド名
- 要求 URI
- プロトコルバージョン。

この API が応答で起動されると、実行時エラーがトリガーされます。

**例：メソッド、要求 URI、プロトコルバージョンのローカル変数の値を設定します。**

### スクリプト

```
M = {}

function M.outbound_INVITE(msg)
    local method, ruri, ver = msg:getRequestLine()
end

return M
```

### メッセージ

```
INVITE sip:1234@10.10.10.1 SIP/2.0
```

### 出力/結果

```
Local variables method, ruri, and version are set accordingly:
method == "INVITE"
ruri == "sip:1234@10.10.10.1"
version == "SIP/2.0"
```

## getRequestUriParameter

`getRequestUriParameter(parameter-name)` returns the URI parameter-value

この関数では、要求 URI パラメータの文字列名が指定された場合、指定された URI パラメータの値を返されます。

- URI パラメータが「tag=value」の形式の場合、等記号の右側にある値が返されます。
- URI パラメータが値のないタグの場合、値は空文字列（例：""）で返されます。
- URI パラメータが最初のヘッダー値にない場合、nil が返されます。
- このメソッドが応答メッセージに対して起動されると、nil が返されます。

**例：ローカル変数を、要求 URI のユーザパラメータの値に設定します。**

### スクリプト

```
M = {}
function M.outbound_INVITE(msg)
    local userparam = msg:getRequestUriParameter("user")
end
return M
```

### メッセージ

```
INVITE sip:1234@10.10.10.1;user=phone SIP/2.0
```

### 出力/結果

ローカル変数 `userparam` が、文字列「phone」に設定されます

## setRequestUri

`setRequestUri(uri)`

このメソッドでは、要求行の要求 URI が設定されます。この API が応答で起動されると、実行時エラーがトリガーされます。

**例：要求 URI を tel:1234 に設定します。**

#### スクリプト

```
M = {}
function M.outbound_INVITE(msg)
  msg.setRequestUri("tel:1234")
end
return M
```

#### メッセージ

```
INVITE sip:1234@10.10.10.1 SIP/2.0
```

#### 出力/結果

```
INVITE tel:1234 SIP/2.0
```

## getResponseLine

`getResponseLine()` returns version, status-code, reason-phrase

このメソッドでは、プロトコルバージョン（文字列）、ステータスコード（整数）、および理由フレーズ（文字列）の3つの値が返されます。この API が要求に対して起動されると、実行時エラーがトリガーされます。

**例：ローカル変数を、プロトコルバージョン、ステータスコード、および理由フレーズの値に設定します。**

#### スクリプト

```
M = {}
function M.outbound_200_INVITE(msg)
  local version, status, reason = msg.getResponseLine()
end
return M
```

#### メッセージ

```
SIP/2.0 200 Ok
.
.
CSeq: 102 INVITE
```

#### 出力/結果

```
Local variables method, ruri, and version are set accordingly:
version == "SIP/2.0"
status == 200
reason == "Ok"
```

## setResponseCode

`setResponseCode(status-code, reason-phrase)`

このメソッドでは、ステータスコードおよび理由フレーズが設定されます。どちらの値にも `nil` を使用できます。値に `nil` が指定されている場合、メッセージに格納されている既存の値が使用されます。この API が要求に対して起動されると、実行時エラーがトリガーされます。

**例：発信 404 応答を 604 応答に変更します。**

#### スクリプト

```
M = {}
function M.outbound_404_INVITE(msg)
    msg:setResponseCode(604, "Does Not Exist Anywhere")
end
return M
```

#### メッセージ

```
SIP/2.0 404 Not Found
.
.
CSeq: 102 INVITE
```

#### 出力/結果

```
SIP/2.0 604 Does Not Exist Anywhere
.
.
CSeq: 102 INVITE
```

## ヘッダーの操作

- [allowHeader](#)
- [getHeader](#)
- [getHeaderValues](#)
- [getHeaderValueParameter](#)
- [getHeaderUriParameter](#)
- [addHeader](#)
- [addHeaderValueParameter](#)
- [addHeaderUriParameter](#)
- [modifyHeader](#)
- [applyNumberMask](#)
- [convertDiversionToHI](#)
- [convertHIToDiversion](#)
- [removeHeader](#)
- [removeHeaderValue](#)

## allowHeader

`allowHeaders` is a Lua table specified by the script writer

Cisco Unified CM が認識していないヘッダーに対するデフォルト動作では、このようなヘッダーを無視します。場合によっては、このようなヘッダーを正規化時に使用したり、ヘッダーを透過的に受け渡したりすることが望ましいこともあります。**allowHeaders** テーブルでヘッダー名を指定することにより、スクリプトを使用して、Cisco Unified CM でのヘッダーの認識が可能になり、スクリプトで正規化のためにローカルで使用したり、透過的に受け渡したりすることは問題なく行えます。

#### 例：

**allowHeaders** は `History-Info` を指定します。指定しない場合は、スクリプトの処理前に、着信 `History-Info` ヘッダーがドロップされます。したがって、**convertHIToDiversion** で有用な結果は得られません。このスクリプトでは、`x-pbx-id` と呼ばれる仮想ヘッダーも許可されます。これは、**allowHeaders** がヘッダー名のテーブルであることを示しています。

#### スクリプト

```
M = {}
M.allowHeaders = {"History-Info", "x-pbx-id"}
function M.inbound_INVITE(msg)
    msg:convertHIToDiversion()
end
return M
```

#### メッセージ

```
INVITE sip:1001@10.10.10.1 SIP/2.0
.
History-Info: <sip:2401@10.10.10.2?Reason=sip;cause=302;text="unconditional">;index=1
History-Info: <sip:1001@10.10.10.1>;index=1.1
```

#### 結果

```
INVITE sip:1001@10.10.10.1 SIP/2.0
.
Diversion: <sip:2401@10.10.10.2>;reason=unconditional;counter=1
History-Info: <sip:2401@10.10.10.2?Reason=sip;cause=302;text="unconditional">;index=1
History-Info: <sip:1001@10.10.10.1>;index=1.1
```

## getHeader

`getHeader(header-name)` returns header-value or nil

このメソッドでは、ヘッダーの文字列名が指定された場合、ヘッダーの値が返されます。同じ名前前の複数ヘッダーについては、値が連結され、カンマで区切られます。

例：ローカル変数を **Allow** ヘッダーの値に設定します。

#### スクリプト

```
M = {}
function M.outbound_INVITE(msg)
    local allow = msg:getHeader("Allow")
end
return M
```

#### メッセージ

```
INVITE sip:1234@10.10.10.1 SIP/2.0
.
To: <sip:1234@10.10.10.1>;
.
```

```
Allow: UPDATE
Allow: Ack,Cancel,Bye,Invite
```

### 出力/結果

```
Local variable allow set to the string "UPDATE,Ack,Cancel,Bye,Invite"
```

## getHeaderValues

getHeaderValues(header-name) returns a table of header values

この関数では、ヘッダーの文字列名が指定された場合、値がインデックス化されたテーブルとして、ヘッダーのカンマ区切りの値が返されます。Lua のインデックス化は 1 から始まります。n 個のヘッダー値がある場合、最初の値はインデックス 1 であり、最後の値はインデックス n です。Lua の **ipairs** 関数では、テーブル全体で繰り返し使用できます

**例：History-Info ヘッダーの値を含むローカル テーブルを作成します。**

### スクリプト

```
M = {}
M.allowHeaders = {"History-Info"}
function M.inbound_INVITE(msg)
    local history_info = msg:getHeaderValues("History-Info")
end
return M
```

### メッセージ

```
INVITE sip:1234@10.10.10.1 SIP/2.0
.
To: <sip:1234@10.10.10.1>;
.
History-Info: <sip:UserB@hostB?Reason=sip;cause=408>;index=1
History-Info: <sip:UserC@hostC?Reason=sip;cause=302>;index=1.1
History-Info: <sip:UserD@hostD>;index=1.1.1
```

### 出力/結果

```
Local variable history_info is set to
{
    "<sip:UserB@hostB?Reason=sip;cause=408>;index=1",
    "<sip:UserC@hostC?Reason=sip;cause=302>;index=1.1",
    "<sip:UserD@hostD>;index=1.1.1"
}

In other words:
history_info[1] == "<sip:UserB@hostB?Reason=sip;cause=408>;index=1"
history_info[2] == "<sip:UserC@hostC?Reason=sip;cause=302>;index=1.1"
history_info[3] == "<sip:UserD@hostD>;index=1.1.1"
```

## getHeaderValueParameter

getHeaderValueParameter(header-name, parameter-name) returns the parameter-value

このメソッドでは、ヘッダーの名前およびヘッダー パラメータが指定された場合、指定されたヘッダー パラメータの値が返されます。同じ名前の複数のヘッダーでは、最初のヘッダー値のみが評価されます。

- ヘッダー パラメータが「tag=value」の形式の場合、等記号の右側にある値が返されます。
- ヘッダー パラメータが値のないタグの場合、値は空文字列（例：""）で返されます。
- ヘッダー パラメータが最初のヘッダー値にない場合、nil が返されます。

**例：ローカル変数を、tag という名前のヘッダー パラメータの値に設定します。**

### スクリプト

```
M = {}
function M.outbound_180_INVITE(msg)
    local totag = msg.getHeaderValueParameter("To", "tag")
end
return M
```

### メッセージ

```
SIP/2.0 180 Ringing
.
To: <sip:1234@10.10.10.1>;tag=32355SIPpTag0114
```

### 出力/結果

```
Local variable totag is set to the string "32355SIPpTag0114"
```

## getHeaderUriParameter

getHeaderUriParameter(header-name, parameter-name) returns the URI parameter-value

このメソッドでは、ヘッダーの文字列名が指定された場合、指定された URI パラメータの値が返されます。同じ名前の複数のヘッダーでは、最初のヘッダー値のみが評価されます。

- URI パラメータが「tag=value」の形式の場合、等記号の右側にある値が返されます。
- URI パラメータが値のないタグの場合、値は空文字列（例：""）で返されます。
- URI パラメータが最初のヘッダー値にない場合、nil が返されます。

**例：ローカル変数を、user という名前の uri パラメータの値に設定します。**

### スクリプト

```
M = {}
function M.outbound_180_INVITE(msg)
    local userparam = msg.getHeaderUriParameter("To", "user")
end
return M
```

### メッセージ

```
SIP/2.0 180 Ringing
.
To: <sip:1234@10.10.10.1;user=phone>;tag=32355SIPpTag0114
```

### 出力/結果

```
Local variable userparam is set to the string "phone"
```

## addHeader

```
addHeader(header-name, header-value)
```

このメソッドでは、ヘッダーの文字列名および値が指定された場合、ヘッダー値のリストの末尾に値が付加されます。

**例：Allow ヘッダーに INFO を追加します。**

### スクリプト

```
M = {}
function M.outbound_INVITE(msg)
    msg.addHeader("Allow", "INFO")
end
return M
```

### メッセージ

```
INVITE sip:1234@10.10.10.1 SIP/2.0
.
To: <sip:1234@10.10.10.1>;
Allow: Ack,Cancel,Bye,Invite
```

### 出力/結果

```
INVITE sip:1234@10.10.10.1 SIP/2.0
.
To: <sip:1234@10.10.10.1>;
Allow: Ack,Cancel,Bye,Invite,INFO
```

## addHeaderValueParameter

```
addHeaderValueParameter(header-name, parameter-name [,parameter-value])
```

このメソッドでは、ヘッダー名、パラメータ名、およびオプションパラメータ値が指定された場合、SIP メッセージで指定されたヘッダーが検索され、指定されたヘッダーパラメータおよびオプションパラメータ値がヘッダーに追加されます。SIP メッセージで、元のヘッダーが変更されたヘッダー値に置き換えられます。指定されたヘッダーに対して複数のヘッダー値がある場合、最初のヘッダー値のみが変更されます。指定されたヘッダーのインスタンスが SIP メッセージにない場合、アクションは実行されません。

**例：発信 Contact ヘッダーに color=blue ヘッダーパラメータを追加します。**

### スクリプト

```
M = {}
function M.outbound_INVITE(msg)
    msg.addHeaderValueParameter("Contact", "color", "blue")
end
return M
```



**メッセージ**

```
INVITE sip:1234@10.10.10.1 SIP/2.0
.
Contact: <sip:1234@10.10.10.2>
```

**出力/結果**

```
INVITE sip:1234@10.10.10.1 SIP/2.0
.
Contact: <sip:1234@10.10.10.12>;color=blue
```

## addHeaderUriParameter

```
addHeaderUriParameter(header-name, parameter-name [,parameter-value])
```

このメソッドでは、ヘッダー名、パラメータ名、およびオプションパラメータ値が指定された場合、SIP メッセージで指定されたヘッダーが検索され、指定されたヘッダー URI パラメータおよびオプション URI パラメータ値が、含まれている URI に追加されます。SIP メッセージで、元のヘッダーが変更されたヘッダー値に置き換えられます。指定されたヘッダーに対して複数のヘッダー値がある場合、最初のヘッダー値のみが変更されます。SIP メッセージに指定されたヘッダーのインスタンスがない場合、または、指定されたヘッダーの最初のインスタンス内に有効な URI が見つからない場合、アクションは実行されません。

**例 : P-Asserted-Identity URI に、user=phone パラメータを追加します。**

**スクリプト**

```
M = {}
function M.inbound_INVITE(msg)
    msg:addHeaderUriParameter("P-Asserted-Identity", "user", "phone")
end
return M
```

**メッセージ**

```
INVITE sip:1234@10.10.10.1 SIP/2.0
.
P-Asserted-Identity: <sip:1234@10.10.10.1>
```

**出力/結果**

```
INVITE sip:1234@10.10.10.1 SIP/2.0
.
P-Asserted-Identity: <sip:1234@10.10.10.1;user=phone>
```

## modifyHeader

```
modifyHeader(header-name, header-value)
```

このメソッドでは、ヘッダーの文字列名および値が指定された場合、既存のヘッダー値が指定された値で上書きされます。これは、`removeHeader` が起動された後で `addHeader` が起動される動作に類似しています。

**例：発信 INVITE メッセージの P-Asserted-Identity ヘッダーから、表示名を削除します。**

### スクリプト

```
M = {}
function M.outbound_INVITE(msg)
    -- Remove the display name from the PAI header
    local pai = msg.getHeader("P-Asserted-Identity")
    local uri = string.match(pai, "<.+>")
    msg.modifyHeader("P-Asserted-Identity", uri)
end
return M
```

### メッセージ

```
INVITE sip:1234@10.10.10.1 SIP/2.0
.
P-Asserted-Identity: "1234" <1234@10.10.10.1>
```

### 出力/結果

```
INVITE sip:1234@10.10.10.1 SIP/2.0
.
P-Asserted-Identity: <1234@10.10.10.1>
```

## applyNumberMask

```
applyNumberMask(header-name, mask)
```

このメソッドでは、ヘッダー名および番号マスクが指定された場合、SIP メッセージで指定されたヘッダーが検索され、指定された番号マスクが、ヘッダー内に含まれている URI に適用されます。URI がヘッダー値から正常に解析された場合、解析された URI のユーザ部分に番号マスクが適用されます。SIP メッセージで、元のヘッダーが変更されたヘッダー値に置き換えられます。指定されたヘッダーに複数のインスタンスがある場合、ヘッダーの最初のインスタンスのみが変更されます。

次のいずれかの条件を満たす場合、アクションは実行されません。

1. 指定されたヘッダーのインスタンスが、SIP メッセージにない場合
2. 指定されたヘッダーの最初のインスタンス内に、有効な URI がない場合
3. 指定されたマスク パラメータが空の文字列の場合

## 番号マスクの適用

マスク パラメータによって、ヘッダー URI のユーザ部に適用される変換が定義されます。マスク パラメータでは、大文字の X でワイルドカード文字が指定されます。たとえば、マスク「+1888XXXXXXX」が指定されている場合、マスクの挿入文字として「X」が使用されます。このマスクが例のユーザ部「4441234」に適用される場合、結果の文字列は「+18884441234」になります。

マスクされるユーザ部に見つかった文字数が、マスクのワイルドカード文字数よりも少ない場合、一番左のワイルドカード文字から「X」が置かれます。前述のマスクを例のユーザ部「1234」に適用すると、結果の文字列は「+1888XXX1234」になります。マスクされるユーザ部に見つかった文字数が、マスクのワイルドカード文字数よりも多い場合、ユーザ部の一番左の文字から切り捨てられます。たとえば、マスク「+1888XXXX」がユーザ部「4441234」に適用される場合、結果の文字列は「+18881234」になります。

**例：着信 INVITE メッセージの P-Asserted-Identity ヘッダーの番号に番号マスクを適用します。**

### スクリプト

```
M = {}
function M.inbound_INVITE(msg)
    msg:applyNumberMask("P-Asserted-Identity", "+1919476XXXX")
end
return M
```

### メッセージ

```
INVITE sip:1234@10.10.10.1 SIP/2.0
.
P-Asserted-Identity: <sip:1234@10.10.10.1>
```

### 出力/結果

```
INVITE sip:1234@10.10.10.1 SIP/2.0
.
P-Asserted-Identity: <sip:+19194761234@10.10.10.1>
```

## convertDiversionToHI

```
convertDiversionToHI()
```

Cisco Unified CM では、リダイレクト情報を処理するための Diversion ヘッダーの送信がサポートされます。たとえば、Cisco Unified CM によってコールが転送される場合について考えます。このような例では、リダイレクト番号の送信のために Cisco Unified CM で使用される Diversion ヘッダーが、SIP メッセージに含まれる場合があります。ただし、多くの SIP サーバでは、この目的で、Diversion ヘッダーの代わりに History-Info ヘッダーが使用されます。この API を使用して、Diversion ヘッダーが History-Info ヘッダーに変換されます。



(注)

一部の実装では、この API の代わりにまたはこの API に加えて、raw ヘッダー API (例: getHeader、addHeader、modifyHeader、removeHeader) の使用が必要です。

**例：発信 INVITE メッセージで、Diversion ヘッダーを History-Info ヘッダーに変換します。次に、Diversion ヘッダーを削除します**

### スクリプト

```
M = {}
function M.outbound_INVITE(msg)
    if msg:getHeader("Diversion")
    then
        msg:convertDiversionToHI()
        msg:removeHeader("Diversion")
    end
end
return M
```

### メッセージ

```
INVITE sip:2400@10.10.10.2 SIP/2.0
.
Diversion: <sip:1002@10.10.10.1>;reason=unconditional;privacy=off;screen=yes
```

**出力/結果**

```
INVITE sip:2400@10.10.10.2 SIP/2.0
.
History-Info: <sip:1002@10.10.10.1?Reason=sip;cause=302;text="unconditional">;index=1
History-Info: <sip:2400@10.10.10.2>;index=1.1
```

## convertHIToDiversio

```
convertHIToDiversio()
```

Cisco Unified CM では、リダイレクト情報を処理するための **Diversio** ヘッダーの受信がサポートされます。たとえば、Cisco Unified CM に到達する前にコールが転送される場合について考えます。このような場合、リダイレクト番号の確立のために Cisco Unified CM で使用される **Diversio** ヘッダーが、SIP メッセージに含まれることがあります。ただし、多くの SIP サーバでは、この目的で、**Diversio** ヘッダーの代わりに **History-Info** ヘッダーが使用されます。



**(注)** 一部の実装では、この API の代わりにまたはこの API に加えて、raw ヘッダー API (例: `getHeader`、`addHeader`、`modifyHeader`、`removeHeader`) の使用が必要な場合があります。



**(注)** **Diversio** ヘッダーは、**History-Info** ヘッダーに「`cause=`」タグが存在する場合にだけ追加されます。以下の例では、**Diversio** ヘッダーが 1 つしかないが、**History-Info** ヘッダーが 2 つある場合、そのうちの 1 つには「`cause=`」タグがあることに注意してください。

**例**

`allowHeaders` では、「**History-Info**」を指定する必要があります。指定しない場合、スクリプトの処理前に、着信 **History-Info** ヘッダーがドロップされます。したがって、`convertHIToDiversio` へのコールで有用な結果は得られません。

**スクリプト**

```
M = {}
M.allowHeaders = {"History-Info"}
function M.inbound_INVITE(msg)
    msg:convertHIToDiversio()
end
return M
```

**メッセージ**

```
INVITE sip:1001@10.10.10.1 SIP/2.0
.
History-Info: <sip:2401@10.10.10.2?Reason=sip;cause=302;text="unconditional">;index=1
History-Info: <sip:1001@10.10.10.1>;index=1.1
```

**結果**

```
INVITE sip:1001@10.10.10.1 SIP/2.0
.
Diversio: <sip:2401@10.10.10.2>;reason=unconditional;counter=1
History-Info: <sip:2401@10.10.10.2?Reason=sip;cause=302;text="unconditional">;index=1
History-Info: <sip:1001@10.10.10.1>;index=1.1
```

## removeHeader

```
removeHeader (header-name)
```

このメソッドでは、ヘッダーの文字列名が指定された場合、メッセージからヘッダーが削除されます。

**例：発信 INVITE メッセージから、Cisco-Guid ヘッダーを削除します。**

```
M = {}
function M.outbound_INVITE(msg)
    msg.removeHeader("Cisco-Guid")
end
return M
```

### メッセージ

```
INVITE sip:1234@10.10.10.1 SIP/2.0
.
P-Asserted-Identity: "1234" <1234@10.10.10.1>
Cisco-Guid: 1234-4567-1234
Session-Expires: 1800
```

### 出力/結果

```
INVITE sip:1234@10.10.10.1 SIP/2.0
.
P-Asserted-Identity: "1234" <1234@10.10.10.1>
Session-Expires: 1800
.
```

## removeHeaderValue

```
removeHeaderValue (header-name, header-value)
```

このメソッドでは、ヘッダーの文字列名およびヘッダー値が指定された場合、メッセージからヘッダー値が削除されます。値がヘッダー値だけだった場合、ヘッダーそのものが削除されます。

**例：発信 INVITE メッセージのサポート ヘッダーから、X-cisco-srtp-fallback を削除します。**

### スクリプト

```
M = {}
function M.outbound_INVITE(msg)
    msg.removeHeaderValue("Supported", "X-cisco-srtp-fallback")
end
return M
```

### メッセージ

```
INVITE sip:1234@10.10.10.1 SIP/2.0
.
P-Asserted-Identity: "1234" 1234@10.10.10.1
Supported: timer, replaces, X-cisco-srtp-fallback
Session-Expires: 1800
```

### 出力/結果

```
INVITE sip:1234@10.10.10.1 SIP/2.0
.
P-Asserted-Identity: "1234" 1234@10.10.10.1
```

```
Supported: timer, replaces
Session-Expires: 1800
```

## SDP の操作

- [getSdp](#)
- [setSdp](#)
- [removeUnreliableSdp](#)

### getSdp

getSdp() returns string

このメソッドでは、Lua 文字列として SDP が返されます。メッセージに SDP が含まれない場合、nil が返されます。

**例：SDP が含まれるローカル変数を確立します（SDP がない場合は nil）。**

#### スクリプト

```
M = {}
function M.outbound_INVITE(msg)
    local sdp = msg:getSdp()
end
return M
```

#### メッセージ

```
INVITE sip:901@rawlings.cisco.com:5080 SIP/2.0
Via: SIP/2.0/TCP 172.18.197.88:5080;branch=z9hG4bK4bae847b2
From: <sip:579@172.18.197.88>;tag=9f12addc-04fc-4f0f-825b-3cd7d6dd8813-25125665
To: <sip:901@rawlings.cisco.com>
Date: Thu, 28 Jan 2010 01:09:39 GMT
Call-ID: cd80d100-b601e3d3-15-58c512ac@172.18.197.88
.
.
Max-Forwards: 69
Content-Type: application/sdp
Content-Length: 214

v=0
o=CiscoSystemsCCM-SIP 2000 1 IN IP4 172.18.197.88
s=SIP Call
c=IN IP4 172.18.197.88
t=0 0
m=audio 24580 RTP/AVP 0 101
a=rtpmap:0 PCMU/8000
a=ptime:20
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-15
```

#### 出力/結果

変数 `sdp` が次の文字列に設定されます。この文字列には、組み込みの復帰改行文字および改行文字（つまり「\r\n」）が含まれていることに注意してください。

```
v=0
```

```

o=CiscoSystemsCCM-SIP 2000 1 IN IP4 172.18.197.88
s=SIP Call
c=IN IP4 172.18.197.88
t=0 0
m=audio 24580 RTP/AVP 0 101
a=rtpmap:0 PCMU/8000
a=ptime:20
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-15

```

## setSdp

```
setSdp(sdp)
```

このメソッドでは、SIP メッセージで指定された文字列を SDP コンテンツ本文として保存します。SIP メッセージの Content-Length ヘッダーは、文字列の長さに自動的に更新されます。

**例：発信 SDP から a=ptime の行を削除します。**

### スクリプト

```

M = {}
function M.outbound_INVITE(msg)
    local sdp = msg:getSdp()
    if sdp
    then
        -- remove all ptime lines
        sdp = sdp:gsub("a=ptime:%d*\r\n", "")
        -- store the updated sdp in the message object
        msg:setSdp(sdp)
    end
end
return M

```

### メッセージ

```

INVITE sip:901@rawlings.cisco.com:5080 SIP/2.0
Via: SIP/2.0/TCP 172.18.197.88:5080;branch=z9hG4bK4bae847b2
From: <sip:579@172.18.197.88>;tag=9f12addc-04fc-4f0f-825b-3cd7d6dd8813-25125665
To: <sip:901@rawlings.cisco.com>
Date: Thu, 28 Jan 2010 01:09:39 GMT
Call-ID: cd80d100-b601e3d3-15-58c512ac@172.18.197.88
.
.
Max-Forwards: 69
Content-Type: application/sdp
Content-Length: 214

v=0
o=CiscoSystemsCCM-SIP 2000 1 IN IP4 172.18.197.88
s=SIP Call
c=IN IP4 172.18.197.88
t=0 0
m=audio 24580 RTP/AVP 0 101
a=rtpmap:0 PCMU/8000
a=ptime:20
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-15

```

## 出力/結果

前述のスクリプト実行後の最後の SIP メッセージは、次のようになります。

```

INVITE sip:901@rawlings.cisco.com:5080 SIP/2.0
Via: SIP/2.0/TCP 172.18.197.88:5080;branch=z9hG4bK4bae847b2
From: <sip:579@172.18.197.88>;tag=9f12addc-04fc-4f0f-825b-3cd7d6dd8813-25125665
To: <sip:901@rawlings.cisco.com>
Date: Thu, 28 Jan 2010 01:09:39 GMT
Call-ID: cd80d100-b601e3d3-15-58c512ac@172.18.197.88
.
.
Max-Forwards: 69
Content-Type: application/sdp
Content-Length: 214

v=0
o=CiscoSystemsCCM-SIP 2000 1 IN IP4 172.18.197.88
s=SIP Call
c=IN IP4 172.18.197.88
t=0 0
m=audio 24580 RTP/AVP 0 101
a=rtpmap:0 PCMU/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-15

```

## removeUnreliableSdp

```
removeUnreliableSdp()
```

このメソッドでは、18X メッセージから SDP が削除されます。SCDP を削除する前に、メッセージでは、PRACK が必要ではないことが確認されます。



(注)

Content-Length ヘッダーへの調整は、この API によって SDP が実際に削除されるときに、自動的に行われます。メッセージに他のコンテンツ本文がない場合、Content-Type ヘッダーは自動的に削除されます。

**例：着信 180 メッセージから SDP を削除します。**

### スクリプト

```

M = {}
function M.inbound_180_INVITE(msg)
    msg.removeUnreliableSdp()
end
return M

```

### メッセージ

```

SIP/2.0 180 Ringing
.
Content-Type: application/sdp
Content-Length: 214

v=0
o=CiscoSystemsCCM-SIP 2000 1 IN IP4 172.18.197.88
s=SIP Call

```



```
c=IN IP4 172.18.197.88
t=0 0
m=audio 24580 RTP/AVP 0 101
a=rtpmap:0 PCMU/8000
a=ptime:20
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-15
```

### 出力/結果

前述のスクリプト実行後の最後の SIP メッセージは、次のようになります。

```
SIP/2.0 180 Ringing
.
Content-Length: 0
```

## コンテンツ本文の操作

- [getBody](#)
- [addBody](#)
- [removeBody](#)

## getBody

`getBody(content-type)` returns the content-body, content-disposition, content-encoding, and content-language of the specified content-type

この関数では、`content-type` の文字列名が指定された場合、`content-body`、`content-disposition`、`content-encoding`、および `content-language` が返されます。`content-disposition`、`content-encoding`、および `content-language` はオプションで、メッセージで指定されていない場合があります。メッセージで特定のヘッダーが指定されていない場合、その値には `nil` が返されます。

メッセージに `content-body` がない場合、返されるすべての項目に `nil` 値が含まれます。

**例：発信 INFO メッセージのコンテンツ情報が含まれるローカル変数を確立します。**

### スクリプト

```
M = {}
function M.inbound_INFO(msg)
  local b = msg.getBody("application/vnd.nortelnetworks.digits")
end
return M
```

### 発信メッセージ

```
INFO sip: 1000@10.10.10.1 SIP/2.0
Via: SIP/2.0/UDP 10.10.10.57:5060
From: <sip:1234@10.10.10.57>;tag=d3f423d
To: <sip:1000@10.10.10.1>;tag=8942
Call-ID: 312352@10.10.10.57
Cseq: 5 INFO
Content-Type: application/vnd.nortelnetworks.digits
Content-Length: 72

p=Digit-Collection
```

```

y=Digits
s=success
u=12345678
i=87654321
d=4

```

### 出力/結果

```

Local variable b is set to the string:

p=Digit-Collection
y=Digits
s=success
u=12345678
i=87654321
d=4

```

## addContentBody

```

addContentBody(content-type, content-body [,content-disposition [,content-encoding
[,content-language]])

```

この API では、`content-type` および `content-body` が指定された場合、メッセージにコンテンツ本文が追加され、`content-length` ヘッダーに必要な変更が行われます。スクリプトでは、オプションで、ディスポジション、エンコード、および言語も指定できます。

**例：発信 UPDATE メッセージに、プレーン テキストのコンテンツ本文を追加します。**

### スクリプト

```

M = {}
function M.outbound_UPDATE(msg)
    msg:addContentBody("text/plain", "d=5")
end
return M

```

### 正規化前のメッセージ

```

UPDATE sip: 1000@10.10.10.1 SIP/2.0
.
Content-Length: 0

```

### 正規化後のメッセージ

```

UPDATE sip: 1000@10.10.10.1 SIP/2.0
.
Content-Length: 3
Content-Type: text/plain
d=5

```

## removeContentBody

```

removeContentBody(content-type)

```

この API では、`content-type` が指定された場合、関連付けられているコンテンツ本文がメッセージから削除され、`content-length` ヘッダーに必要な変更が行われます。

例：発信 UPDATE メッセージから text/plain コンテンツ本文を削除します。

#### スクリプト

```
M = {}
function M.outbound_UPDATE(msg)
    msg.removeContentBody("text/plain")
end
return M
```

#### 正規化前のメッセージ

```
UPDATE sip: 1000@10.10.10.1 SIP/2.0
.
Content-Length: 3
Content-Type: text/plain

d=5
```

#### 正規化後のメッセージ

```
UPDATE sip: 1000@10.10.10.1 SIP/2.0
.
Content-Length: 0
```

## メッセージのブロック

- [block](#)

### block

```
block()
```

このメソッドでは、着信または発信の信頼できない 18X メッセージをブロックできます。着信のブロックでは、メッセージを受信しなかったものとして CUCM が動作します。発信メッセージのブロックでは、CUCM によってネットワークにメッセージが送信されません。



(注) PRACK が有効で、メッセージが実際は信頼できる 18X の場合、CUCM では block() への呼び出しは事実上無視します。メッセージはブロックされず、エラー SDI トレースが生成されます。

例：発信 183 メッセージをブロックします。

#### スクリプト

```
M = {}
function M.outbound_183_INVITE(msg)
    msg.block()
end
return M
```

#### メッセージ

```
SIP/2.0 183 Progress
Via: SIP/2.0/TCP 172.18.199.186:5060;branch=z9hG4bK5607aa91b
From: <sip:2220@172.18.199.186>;tag=7cae99f9-2f13-4a4d-b688-199664cc38a4-32571707
```

```
To: <sip:2221@172.18.199.100>;tag=3affe34f-6434-4295-9a4b-c1fe2b0e03b6-21456114
.
.
```

### 出力/結果

```
The 183 message is not sent to the network.
```

## 透過性

- [getPassThrough](#)

## getPassThrough

`getPassThrough()` returns a per message pass through object

このメソッドでは、SIP パススルー オブジェクトが返されます。これは、着信メッセージに対して起動できます。パススルー オブジェクトに追加される情報は、他のコール レッグで生成された発信メッセージに自動的に結合されます。この情報の自動結合は、(必要に応じて) 発信の正規化の前に発生します。

パススルー オブジェクトの使用および例については、[SIP パススルー API](#)を参照してください。

## Per-Dialog コンテキストの管理

- [getContext](#)

## getContext

`getContext()` returns a per call context

このメソッドでは、コールごとのコンテキストが返されます。コンテキストは Lua テーブルです。このスクリプト作成者は、ダイアログの継続中に使用される、多様なメッセージ ハンドラでのコンテキストからデータを保存および取得することができます。スクリプト作成者は、ダイアログの最後でコンテキストの解放を考慮する必要はありません。コンテキストは、ダイアログの終了時に、Cisco Unified CM によって自動的に解放されます。

### 例:

このスクリプトでは、コンテキストを使用して、INVITE で受信した **History-Info** ヘッダーが保存されます。その INVITE への応答が送信されると、保存されたヘッダーが取得され、発信応答に追加されます。「clear」パラメータを使用して、後続の reINVITE 応答にこれらのヘッダーが追加されることを防止します。

### スクリプト

```
M = {}
M.allowHeaders = {"History-Info"}
function M.inbound_INVITE(msg)
    local history = msg.getHeader("History-Info")
    if history
```

```

    then
        msg:convertHIToDiversion()
        local context = msg:getContext()
        if context
        then
            context["History-Info"] = history
        end
    end
end

local function includeHistoryInfo(msg, clear)
    local context = msg:getContext()
    if context
    then
        local history = context["History-Info"]
        if history
        then
            msg:addHeader("History-Info", history)

            if clear
            then
                context["History-Info"] = nil
            end
        end
    end
end

function M.outbound_18X_INVITE(msg)
    includeHistoryInfo(msg)
end

function M.outbound_200_INVITE(msg)
    includeHistoryInfo(msg, "clear")
end

return M

```

### 正規化前のメッセージ

```

INVITE sip:1001@10.10.10.1 SIP/2.0
.
History-Info: <sip:2401@10.10.10.2?Reason=sip;cause=302;text="unconditional">;index=1
History-Info: <sip:1001@10.10.10.1>;index=1.1

SIP/2.0 180 Ringing
.
Content-Length: 0

SIP/2.0 200 OK
.
Content-Type: application/sdp

```

### 正規化後のメッセージ

```

INVITE sip:1001@10.10.10.1 SIP/2.0
.
Diversion: <sip:2401@10.10.10.2>;reason=unconditional;counter=1
History-Info: <sip:2401@10.10.10.2?Reason=sip;cause=302;text="unconditional">;index=1
History-Info: <sip:1001@10.10.10.1>;index=1.1

SIP/2.0 180 Ringing
.
Content-Length: 0

```

```
History-Info: <sip:2401@10.10.10.2?Reason=sip;cause=302;text="unconditional">;index=1,
<sip:1001@10.10.10.1>;index=1.1
```

```
SIP/2.0 200 OK
```

```
.
```

```
Content-Type: application/sdp
```

```
History-Info: <sip:2401@10.10.10.2?Reason=sip;cause=302;text="unconditional">;index=1,
<sip:1001@10.10.10.1>;index=1.1
```

```
.
```

## ユーティリティ

- [isInitialInviteRequest](#)
- [isReInviteRequest](#)
- [getUri](#)

### isInitialInviteRequest

`isInitialInviteRequest()` returns true or false

このメソッドでは、メッセージが要求であり、メソッドが INVITE であり、To ヘッダーにタグパラメータがない場合に、真が返されます。これ以外の場合は、偽が返されます。

#### 例：

発信 INVITE が初期 INVITE かどうかに基づいて、ローカル変数を設定します。この例では、INVITE は初期 INVITE です。したがって、この場合の INVITE については、値は真になります。

#### スクリプト

```
M = {}
function M.outbound_INVITE(msg)
    local isInitialInvite = msg.isInitialInviteRequest()
end
return M
```

#### メッセージ

```
INVITE sip:1234@10.10.10.1 SIP/2.0
.
To: <sip:1234@10.10.10.1>
```

#### 出力/結果

```
Local variable isInitialInvite set to 'true'
```

#### 例：

発信 INVITE が初期 INVITE かどうかに基づいて、ローカル変数を設定します。この例では、INVITE は初期 INVITE ではありません。したがって、この場合の INVITE については、値は偽になります。

#### スクリプト

```
M = {}
```

```
function M.outbound_INVITE(msg)
  local isInitialInvite = msg:isInitialInviteRequest()
end
return M
```

**メッセージ**

```
INVITE sip:1234@10.10.10.1 SIP/2.0
.
To: <sip:1234@10.10.10.1>;tag=1234
```

**出力/結果**

```
Local variable isInitialInvite set to 'false'.
```

## isReInviteRequest

```
isReInviteRequest() returns true or false
```

このメソッドでは、メッセージが要求であり、メソッドが INVITE であり、To ヘッダーにタグパラメータが 1 つある場合に、真が返されます。これ以外の場合は、偽が返されます。

**例:**

発信 INVITE が reINVITE かどうかに基づいて、ローカル変数を設定します。この例では、INVITE は reINVITE です。したがって、この場合の INVITE については、値は真になります。

**スクリプト**

```
M = {}
function M.outbound_INVITE(msg)
  local isReInvite = msg:isReInviteRequest()
end
return M
```

**メッセージ**

```
INVITE sip:1234@10.10.10.1 SIP/2.0
.
To: <sip:1234@10.10.10.1>;tag=112233445566
.
```

**出力/結果**

```
Local variable isReInvite set to 'true'.
```

**例:**

発信 INVITE が reINVITE かどうかに基づいて、ローカル変数を設定します。この例では、INVITE は reINVITE ではありません。したがって、この場合の INVITE については、値は偽になります。

**スクリプト**

```
M = {}
function M.outbound_ANY(msg)
  local isReInvite = msg:isReInviteRequest()
end
return M
```

**メッセージ**

```
CANCEL sip:1234@10.10.10.1 SIP/2.0
```

### 出力/結果

```
Local variable isReInvite set to 'false'.
```

## getUri

```
getUri(header-name) returns a string or nil
```

このメソッドでは、ヘッダー名が指定された場合、SIP メッセージで指定されたヘッダーが検索され、そのヘッダーからの URI の取得が試行されます。指定されたヘッダーに複数のインスタンスがある場合、ヘッダーの最初のインスタンスの URI が返されます。SIP メッセージに指定されたヘッダーのインスタンスがない場合、または、指定されたヘッダーの最初のインスタンス内に有効な URI が見つからない場合、Lua nil が返されます。

**例：ローカル変数を、P-Asserted-Identity ヘッダーの URI に設定します。**

### スクリプト

```
M = {}  
function M.inbound_INVITE(msg)  
    local uri = msg:getUri("P-Asserted-Identity")  
end  
return M
```

### メッセージ

```
INVITE sip:1234@10.10.10.1 SIP/2.0  
.  
P-Asserted-Identity: <sip:1234@10.10.10.1>  
.
```

### 出力/結果

```
Local variable uri is set to "sip:1234@10.10.10.1"
```