CHAPTER 3

SIP メッセージの API

Lua スクリプト環境では、メッセージを操作できる API のセットが提供されます。これらの API について、次の項で説明します。

要求行または応答行での操作

getRequestLine

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

このメソッドでは、次の 3 つの値が返されます。

- 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状態で、指定された URI パラメータの値が返されます。

- 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で起動された場合、nil が返されます。

例：要求 URI で、ローカル変数をユーザ 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）が返されます。この API が要求で起動されると、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。いずれの値にも、nil を設定できます。値に nil が設定された場合、メッセージに保存されている既存値が使用されます。この API が要求で起動されると、実行時エラーがトリガーされます。

例：発信の 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
```

認識しないヘッダーに関連する Cisco Unified CM のデフォルト動作では、このようなヘッダーは無視されます。一部の場合、このようなヘッダーは正規化中に使用されるか、または、ヘッダーが透過的に渡される必要があります。**allowHeaders** テーブルでヘッダー名を使用することによって、スクリプトにより、Cisco Unified CM でヘッダーを認識できるようになり、正規化の場合はスクリプトでローカルに使用されるか、または透過的に渡されます。

例：

allowHeaders によって、`History-Info` が指定されます。指定がない場合、スクリプトの処理前に、着信 History-Info ヘッダーはドロップされます。したがって、**convertHIToDiversion** では、役に立つ結果は生成されません。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状態で、ヘッダーの値が返されます。同じ名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状態で、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 の索引は 1 に基づいています。ヘッダー値 n がある場合、最初の値はインデックス 1 にあり、最後の値はインデックス n にあります。Lua の **ipairs** 機能では、テーブル全体で繰り返し使用できます。

例：History-Info ヘッダーの値が含まれるローカル 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、ヘッダーおよびヘッダー パラメータの名前が指定された状態で、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状態で、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、ヘッダーおよび値の文字列名が指定された状態で、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状態で、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複数のヘッダー値がある場合、最初のヘッダー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、アクションは実行されません。

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

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状態で、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複数のヘッダー値がある場合、最初のヘッダー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、ヘッダーおよび値の文字列名が指定された状態で、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似ています。

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状態で、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 に追加されます。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。

次の条件のいずれかが満たされない場合、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、最も左にあるワイルド カード文字は「X」で残されます。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、ユーザ部の最も左にある文字が切り捨てられます。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 ヘッダーで、番号マスクに番号を適用します。

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。このようなインスタンスでは、リダイレクト番号の送信に 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
```

convertHIToDiversion

```
convertHIToDiversion()
```

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。

（注） 一部の実装では、この API の代わりまたはこの 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。

例

allowHeaders では、「History-Info」を指定する必要があります。指定がない場合、スクリプトの処理前に、着信 History-Info ヘッダーはドロップされます。したがって、convertHIToDiversion の呼び出しによっては、役に立つ結果は生成されません。

スクリプト

```
M = {}
M.allowHeaders = {"History-Info"}
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
```

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状態で、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状態で、メッセージからヘッダー値が削除されます。値にヘッダー値のみがあった場合、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

```
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（SDP がない場合は nil）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.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（例：「\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 Content 本体として保存します。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

前述のスクリプトの実行後の最後の 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 Content 本体がない場合、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

前述のスクリプトの実行後の最後の SIP メッセージは、次のようになります。

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

Content 本体の操作

getContentBody

```
getContentBody(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状態で、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 Content 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:getContentBody("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 が指定された状態で、メッセージに Content 本体が追加され、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、および言語が提供される場合があります。

例：発信 UPDATE メッセージに、プレーン テキストの Content 本体を追加します。

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 が指定された状態で、関連付けられている Content 本体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 メッセージからテキスト / プレーン Content 本体を削除します。

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かのように CUCM が動作します。発信のブロックでは、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可能な 18X の場合、CUCM では block() への呼び出しは事実上無視されます。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、ダイアログの動作中に使用される多様なメッセージ ハンドラで、コンテキストにデータを保存し、コンテキストからデータを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え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、コンテキストが使用され、INVITE で受信した History-Info ヘッダーが保存されます。INVITE 送信の応答時に、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」パラメータが使用され、後続の reINVITE 応答へのこれらのヘッダーの追加が防止されます。

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

```
isInitialInviteRequest() returns true or false
```

このメソッドでは、メッセージが要求された場合、メソッドが INVITE の場合、および、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、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 ではなく、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、メッセージが要求された場合、メソッドが INVITE の場合、および、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、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 ではなく、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

ローカル変数 isReInvite が「false」に設定されます。

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状態で、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：P-Asserted-Identity ヘッダーで、ローカル変数を 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"
```