HP Route Analytics Management System

バージョン : 5.20

開発者用ガイド

Manufacturing Part Number: BA129-99026
Document Release Date: 2007 年 5 月
Software Release Date: 2007 年 5 月

ご注意

1. 本書に記載した内容は、予告なしに変更することがあります。
2. 当社は、本書に関して特定目的の市場性と適合性に対する保証を含む一切の保証をいたしかねます。
3. 当社は、本書の記載事項の誤り、またはマテリアルの提供、性能、使用により発生した損害については責任を負いかねますのでご了承ください。
4. 本製品パッケージとして提供した本書、CD-ROM などの媒体は本製品用だけにお使いください。プログラムをコピーする場合はバックアップ用だけにしてください。プログラムをそのままの形で、あるいは変更を加えて第三者に販売することは固く禁じられています。

Restricted Rights Legend

Copyright Notices

Trademark Notices

サポート情報

次の HP ソフトウェア サポート Web サイトにアクセスできます。

http://support.openview.hp.com/support.jsp

HP ソフトウェアサポート オンラインでは、対話型の技術支援ツールに素早く効率的にアクセスいただけます。サイトのサポート範囲は次のとおりです。

- マニュアル類の検索
- サポートケースの登録とトラッキング、およびエンハンスメント要求 (英語)
- ソフトウェアパッチのダウンロード
- サポート契約の管理
- HP サポートの連絡先の検索
- 利用可能なサービスの確認
- フォーラムへの参加
- ソフトウェアトレーニングの確認と登録

各種サポートのご利用の際は、ほとんどの場合、HP Passport ユーザーとしてご登録いただき、ログインしていただく必要があります。また、サポート契約も必要です。

アクセスレベルと HP Passport に関する詳細は以下を参照してください。

http://support.openview.hp.com/new_access_levels.jsp

原 典

本書は『HP Route Analytics Management System Developer's Guide』(HP Part No. BA129-90026) を翻訳したものです。

目次

1 はじめに 13

2 クエリーの設定と使用方法 15

クエリーを受け付けるように RAMS を設定する方法 16

クエリーの使用方法 18

エラーコード 21

警報値 22

クエリーのデータ構造体 23

非 MP/VPN 呼び出しのデータ構造体 23

MP/VPN 呼び出しのデータ構造体 24

3 リエントラントクエリーの使用方法 27

4 XML-RPC クエリー 31

api_del_watch_list 32

入力パラメータ 32

出力の構造 32

例 33

サンプル出力 33

api_get_alert_destination_info 34

入力パラメータ 34

出力の構造 34

例 35

サンプル出力 36

api_get_watch_list 37

入力パラメータ 37

出力の構造 37

例 38
サンプル出力 39
api_link_list 40
入力パラメータ 40
出力の構造 40
例 41
サンプル出力 42
api_list_a_route 43
入力パラメータ 43
出力の構造 44
例 45
サンプル出力 46
api_list_a_route_ECMP 47
入力パラメータ 47
出力の構造 47
例 49
サンプル出力 50
api_mp_close_handle 51
入力パラメータ 51
出力の構造 51
例 / サンプル 51
api_mp_events 52
入力パラメータ 52
出力の構造 53
例 54
サンプル出力 55
api_mp_events_handle 56
入力パラメータ 56
出力の構造 56
例 / サンプル 56
api_mp_osi_routes 57
入力パラメータ 57
出力の構造 58

例 59
サンプル出力 61
api_mp_osi_routes_handle 63
入力パラメータ 63
出力の構造 63
例 / サンプル 63
api_mp_links 64
入力パラメータ 64
出力の構造 64
例 66
サンプル出力 67
api_mp_list_handle 68
入力パラメータ 68
出力の構造 68
api_mp_list_paths 69
入力パラメータ 69
出力の構造 69
例 71
サンプル出力 72
例 (OSI ネットワーク) 74
サンプル出力 (OSI ネットワーク) 76
api_mp_prefixes_multi_origin 78
入力パラメータ 78
出力の構造 78
例 80
サンプル出力 82
api_mp_prefixes_multi_origin_handle 84
入力パラメータ 84
出力の構造 84
例 / サンプル 84
api_mp_routes 85
入力パラメータ 85
出力の構造 85

例 87
サンプル出力 88
api_mp_routes_handle 89
入力パラメータ 89
出力の構造 89
例 / サンプル 89
api_mp_routers 90
入力パラメータ 90
出力の構造 90
例 92
サンプル出力 93
api_prefix_events 94
入力パラメータ 94
出力の構造 94
例 96
サンプル出力 97
api_prefix_list 98
入力パラメータ 98
出力の構造 98
例 99
サンプル出力 100
api_prefix_list_filtered 101
入力パラメータ 101
出力の構造 101
例 102
サンプル出力 103
api_prefix_list_multi_orig 104
入力パラメータ 104
出力の構造 104
例 105
サンプル出力 106
api_prefix_list_same 108
入力パラメータ 108

出力の構造 108
例 109
サンプル出力 110
api_resource_status 111
入力パラメータ 111
出力の構造 111
例 113
サンプル出力 114
api_router_events 115
入力パラメータ 115
出力の構造 115
例 116
サンプル出力 117
api_router_list 119
入力パラメータ 119
出力の構造 119
例 120
サンプル出力 121
api_router_summarizable 122
入力パラメータ 122
出力の構造 122
例 123
サンプル出力 124
api_system_health 125
入力パラメータ 125
出力の構造 125
例 126
サンプル出力 127
api_vpn_cust_rt_list 129
入力パラメータ 129
出力の構造 129
例 130
サンプル出力 131

api_vpn_customer_pe_participation 132
入力パラメータ 132
出力の構造 132
例 134
サンプル出力 135
api_vpn_customer_pe_list 136
入力パラメータ 136
出力の構造 137
例 138
サンプル出力 139
api_vpn_customer_reachability 140
入力パラメータ 140
出力の構造 140
例 142
サンプル出力 143
api_vpn_customer_reachability_by_peer 144
入力パラメータ 144
出力の構造 144
例 146
サンプル出力 148
api_vpn_route_target_pe_participation 149
入力パラメータ 149
出力の構造 149
例 151
サンプル出力 152
api_vpn_route_target_pe_list 153
入力パラメータ 153
出力の構造 153
例 155
サンプル出力 156
api_vpn_route_target_reachability 157
入力パラメータ 157
出力の構造 157

例 159
サンプル出力 160
api_vpn_route_target_reachability_by_peer 162
入力パラメータ 162
出力の構造 162
例 164
サンプル出力 166
api_vpn_routes 167
入力パラメータ 167
出力の構造 167
例 169
サンプル出力 170
api_vpn_routes_handle 171
入力パラメータ 171
出力の構造 171
例 / サンプル 171

索引 173

1　はじめに

API (アプリケーションプログラミングインタフェース) を使って RAMS クエリーを作成することができます。RAMS クエリーは、XML-RPC (拡張マークアップ言語用リモートプロシージャコール) から発行される特殊なクエリーです。

通常、これらのクエリーは、C、Java、Perl のようなコンピュータ言語で書かれたコンピュータプログラムから発行されます。本書では、Perl スクリプト言語で書かれた例を説明します。

コンピュータプログラムからクエリーを実行すると、以下の処理が可能です。

- RAMS から具体的なルート分析情報を取得する。
- XML-RPC をサポートする他のツールに RAMS を統合する。

2　クエリーの設定と使用方法

本章では、RAMS クエリーの設定方法と使用方法について説明します。プログラムでこれらのクエリーを使用するには、適切な XML-RPC ライブラリまたはパッケージとリンクする必要があります。

注記　XML-RPC では大文字と小文字が区別されます。

詳細は、**http://www.xmlrpc.com** を参照してください。

クエリーを受け付けるように RAMS を設定する方法

クエリーを使用する前に、クエリーを受け付けるように RAMS を設定する必要があります。複数の Route Recorder と集中型の Modeling Engine を使用して開発を行う場合は、クエリーを有効にするときに以下の推奨事項を参考にしてください。

- 警報と監視リストについては、複数の Route Recorder からの情報を必要とする警報 (たとえば、Route Change) を除き、デスティネーションの Route Recorder でクエリーを有効にします。
- ネットワーク全体の情報については、集中型の Modeling Engine でクエリーを有効にします。
- Route Recorder の領域またはプロトコルに固有の情報については、その Route Recorder でクエリーを有効にします。

クエリーを有効にするには、以下の手順を実行します。

1. RAMS または集中型の Modeling Engine の [Home] ページで [**Administration**] をクリックし、左のナビゲーションバーの [**Queries**] をクリックします。

   図 1 に示すように [Queries] ページが表示されます。

2. [**Enable queries**] を選択します。
3. パスワードを入力し、確認用のパスワードも入力します。パスワードには 1 ～ 8 字の英数字を使用し、大文字と小文字と区別して指定します。Null 文字、空白、アンダースコアは使用できません。
4. [**Update**] をクリックします。

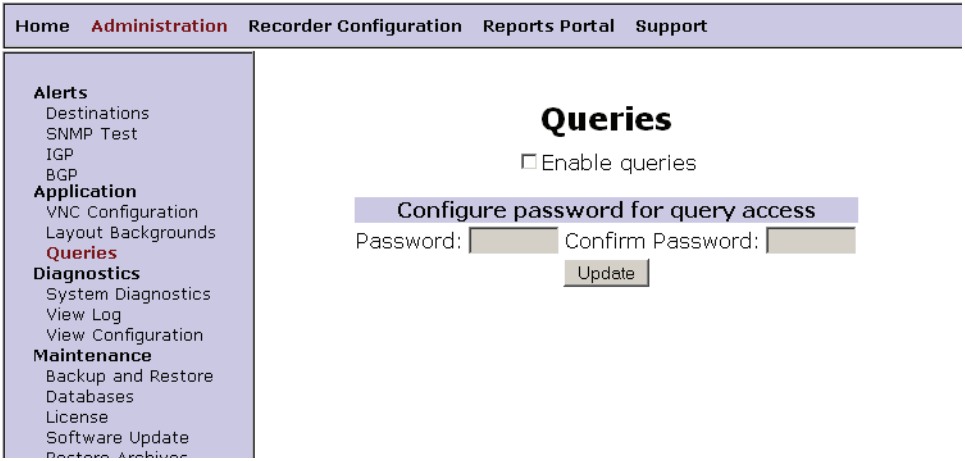

図 1　[Queries] ページ

クエリーの使用方法

第 3 章では、表 1 に示す XML-RPC 呼び出しの入力パラメータと結果を示します。各呼び出しのメソッド名は、接頭辞「RouteAnalyzer.」と、表に示すクエリー名で構成されます。api_mp_ で始まるクエリーは、IGP および BGP の両プロトコルドメインからデータを取得するために使用できます。api_list_a_route は、IGP および BGP の両プロトコルドメインを交差するパスに使用します。api_vpn_ で始まる呼び出しは、BGP/MPLS VPN プロトコルドメインにのみ使用します。残りは、IGP プロトコルドメインにのみ使用します。

> **注記** XML ではセパレータとして「<」と「>」を使用し、改行文字は使用しませんが、サンプルのクエリープログラムで呼び出される Dumper 関数を使用すると、XML を読みやすい形式に変換することができます。第 3 章に示すサンプル出力では、この読みやすい形式を使用します。Dumper 関数は、「Data」という標準 Perl パッケージにあります。

表 1　クエリー呼び出しと説明

クエリー	説明	ページ
api_del_watch_list	指定された警報の監視リストを削除します。	32
api_get_alert_destination_info	指定された警報の宛先に関する情報を返します。	34
api_get_watch_list	指定された警報の監視リストを返します。	37
api_link_list	指定されたネットワークのリンクリストを返します。	40
api_list_a_route	パス (経路) のメトリック総数とリンクリストを返します。	43
api_list_a_route_ECMP	すべての ECMP パスを対象に、パス (経路) のメトリック総数とリンクリストを返します。	47
api_mp_close_handle	このクエリーは、生成済みのレポートハンドルを取得し、レポートを閉じて、レポートが使用しているメモリとリソースを解放します。	51
api_mp_events	2 つの時刻の間に発生したマルチプロトコルネットワークイベントのリストを返します。	52
api_mp_events_handle	2 つの時刻の間に発生したマルチプロトコルネットワークイベントのリストをユーザー指定のサイズに分けて返します。	56

クエリー	説明	ページ
api_mp_links	マルチプロトコルネットワークのフィルター条件を満たすリンクのリストを返します。	64
api_mp_list_handle	レポートのうち、ユーザー指定のエントリから始まるユーザー指定の数のエントリを返します。	68
api_mp_routes	マルチプロトコルネットワークのプレフィックス経路のうち、フィルター条件を満たす経路のリストを返します。	85
api_mp_routes_handle	マルチプロトコルネットワークのプレフィックス経路のうち、フィルター条件を満たす経路のリストをユーザー指定のサイズに分けて返します。	89
api_mp_routers	マルチプロトコルネットワークのルーターのうち、フィルター条件を満たすルーターのリストを返します。	90
api_prefix_events	指定されたプレフィックスのイベントリストを返します。	94
api_prefix_list_filtered	指定されたルーターのプレフィックスリストを返します。	101
api_prefix_list_multi_orig	指定されたネットワークの複数のルーターによって通知されたプレフィックスのリストを返します。	104
api_prefix_list_same	同時に複数のルーターによって通知されたすべてのプレフィックスのリストを返します。	108
api_resource_status	アプライアンスのメモリ、ディスク、スワップ領域の現状を返します。	111
api_router_events	指定されたルーターに関連するイベントのリストを返します。	115
api_router_list	指定されたネットワークのルーターのリストを返します。	119
api_router_summarizable	集約可能な複数のプレフィックスを通知しているルーターのリストを返します。	122

クエリー	説明	ページ
api_system_health	ネットワーク内の Route Explorer システムと Traffic Explorer システムの稼働状態を返します。設定された各記録プロセスの記録ステータスとプロセスのデータベースの書き込みステータスも含まれます。	125
api_vpn_cust_rt_list	VPN カスタマ名と RT (Route Target) のマッピングリストを返します。	129
api_vpn_customer_pe_participation	各 VPN カスタマに関連する PE の統計を返します。	132
api_vpn_customer_pe_list	指定された VPN カスタマに関連する PE のリストを返します。	136
api_vpn_customer_reachability	各 VPN カスタマの到達可能性統計を返します。	140
api_vpn_customer_reachability_by_peer	指定された VPN カスタマの各 PE の到達可能性統計を返します。	144
api_vpn_route_target_pe_participation	指定されたネットワークの各 RT (Route Target) に関連する PE の統計を返します。	149
api_vpn_route_target_pe_list	指定された RT (Route Target) に関連する PE ルーターとその VPN 状態のリストを返します。	153
api_vpn_route_target_reachability	指定されたネットワークの各 RT (Route Target) の到達可能性統計を返します。	157
api_vpn_route_target_reachability_by_peer	指定された RT (Route Target) の各 PE の到達可能性統計を返します。	162
api_vpn_routes	指定されたネットワークの VPN 経路リストを返します。	167
api_vpn_routes_handle	ネットワーク内で通知されているすべてのプレフィックスのリストを返します。	171

エラーコード

RAMS で使用されるエラーコードを表 2 に示します。

表 2　XML-RPC のエラーコード

コード	説明
200	データベース名が無効。
201	トポロジ名が無効。
202	時刻が無効。
203	メモリ割り当てエラー。
204	パスワードの誤り。
205	フィルター式が無効。
206	レポート名の形式が無効。
207	レポート名が無効。
208	VPN カスタマが文字列でない。
209	VPN カスタマが無効。
210	レポートハンドルが無効。
211	RT (Route Target) の形式が無効。
212	RT (Route Target) が無効。
213	監視リストが存在しない。
214	要求の IP アドレス構造体が無効。
215	ソースルーターが存在しない。
216	パスの長さが 0。
219	トラップ名が無効。
220	データベースからルーティングイベントをロードできない。
221	システムリソース情報を処理できない。
222	1 つ以上のデータベースに接続できない。

警報値

表 3 に警報値と対応する警報名を示します。

表 3　警報値と警報名

警報値	警報名
AdjLost	Adjacency Lost
AdjEst	Adjacency Established
LnkFlap	Adjacency Flap
RtChange	Prefix Change
RtOrigChange	Prefix Origination Change
PrefixFlap	Prefix Flap
RtFlap	Route Change
Event	Routing Event
EChurn	Excess Churn
PeerChange	Peer Change
BgpPrefixDrought	Prefix Drought
BgpPrefixFlood	Prefix Flood
BgpRouteFlap	Route Flap
BgpLostRedund	Lost Redundancy
BgpLostPeer	Lost Peer
BgpEstPeer	Established Peer
BgpAcqRedund	Acquired Redundancy
BgpASPathLonger	AP Path Longer
BgpDownToOnePath	Down to one path
BgpDownToZeroPaths	Down To Zero Paths
RtrReachability	Customer Reachability by PE
CustReachability	Customer Reachability
CustPrivacy	Customer PE Participation
RtrIntrusion	New Customer PE

クエリーのデータ構造体

各呼び出しの入力パラメータおよび出力結果に使用されるデータ構造体には、いくつかの種類があります。以下のリストでは、各形式で使用される一般的な構造体 (ルーター、リンク、プレフィックスなど) 間の相違を示すとともに、構造体の要素のデータ型を示します。このリストは、新しいマルチプロトコル (MP)/VPN 呼び出しのデータ構造体と、古い非 MP/VPN 呼び出しのデータ構造体の 2 種類に分かれています。

非 MP/VPN 呼び出しのデータ構造体

非 MP/VPN 呼び出しで使用される一般的な構造体は以下のとおりです。

- IP 構造体 : 以下のいずれかです。
  - ip4_addr (IP4 アドレス): 文字列
  - ip6_addr (IP6 アドレス): 文字列
- プレフィックス構造体 :
  - masklen (マスク長): 整数
  - ip_addr (IP アドレス): IP 構造体
- ルーター構造体 :
  - ip_addr (IP アドレス): IP 構造体
  - maskLen (マスク長): 整数
  - name (名前): 文字列
  - nodeType (ノードのタイプ): 文字列
  - nodeProto (ノードのプロトコル): 文字列
  - nodeArea (ノードの領域): 文字列
  - nodeState (ノードの状態): 文字列
- インタフェース構造体 :
  - source (ソース): IP 構造体
  - destination (デスティネーション): IP 構造体
  - metric (メトリック): 整数
  - bw (帯域幅): double (Kbps 単位)

- delay (遅延): double (マイクロ秒単位)
- state (状態): 整数

- リンク構造体 :
  - source (ソース): ルーター構造体
  - destination (デスティネーション): ルーター構造体
  - interfaces (インタフェース): インタフェース構造体の配列

MP/VPN 呼び出しのデータ構造体

MP/VPN 呼び出しで使用される一般的な構造体は以下のとおりです。

- MP IP 構造体 :
  - ip4_addr (IP4 アドレス): 文字列
- MP プレフィックス構造体 :
  - masklen (マスク長): 整数
  - ip_addr (IP アドレス): MP IP 構造体
- MP 状態構造体 :
  - down (ダウン): 文字列
  - inBaseline (基準値): 文字列 (要求された場合)
- MP ルーター構造体 :
  - name (名前): 文字列 (ルーター名が存在する場合)
  - ipaddr (IP アドレス): MP IP 構造体 (ルーターに IP がある場合)
  - type (タイプ): 文字列
  - sysid (システム ID): 文字列 (プロトコルが ISIS の場合)
  - protoType (プロトコルタイプ): 文字列 (プロトコルが ISIS の場合)
  - model (モデル): 文字列 (プロトコルが EIGRP の場合)
  - softwareVersion (ソフトウェアバージョン): 文字列 (プロトコルが EIGRP の場合)
- MP リンク構造体 :
  - srcNode (ソースノード): MP ルーター構造体
  - dstNode (デスティネーションノード): MP ルーター構造体

— state (状態): MP 状態構造体 (基準値なし)

- BGP 属性構造体 :
  - asPath (AS Path 属性): 文字列 (属性がある場合)
  - origin (Origin 属性): 文字列 (属性がある場合)
  - localPref (Local-Pref 属性): 整数 (属性がある場合)
  - nextHop (Next Hop): 文字列 (属性がある場合)
  - originator (Originator 属性): 文字列 (属性がある場合)
  - clusterList (Cluster List 属性): 文字列 (属性がある場合)
  - aggregator (Aggregator 属性): (属性がある場合)
    - ipaddr (IP アドレス): 文字列
    - as (AS): 整数
  - atomic (Atomic Aggregate 属性): ブール (属性がある場合)
  - extCommunities (Extended Community 属性): 文字列 (属性がある場合)
  - mpReachabililtyNextHop: 文字列 (属性がある場合)
- LS 属性構造体 :
  - metric (メトリック): 整数
  - metricType (Metric Type): 文字列
  - forwardAddr: 文字列 (属性がある場合)
- EIGRP 属性構造体 :
  - metricBW: 整数 (帯域幅の逆数、Bps 単位、$2.56 * 10^{12}$ をかける)
  - metricDelay: 整数 (10 マイクロ秒 * 256 単位)
  - metricType (Metric Type): 文字列
  - ifname: 文字列 (属性がある場合)
- 静的属性構造体 :
  - nextHops: 以下の属性の配列
  - nextHop (Next Hop): 文字列
- トポロジ構造体 :
  - fullName (完全名): 文字列
  - protocol (プロトコル): 文字列

MP/VPN 呼び出しと非 MP/VPN 呼び出しの両方で使用される構造体は以下のとおりです。

- バージョン構造体 :
  - appliance_version (アプライアンスのバージョン): 文字列
  - software_version (ソフトウェアバージョン): 文字列

3　リエントラントクエリーの使用方法

クエリー api_mp_events、api_mp_routes、api_vpn_routes には、以下のリエントラントバージョンがあります。

- api_mp_events_handle
- api_mp_osi_routes_handle
- api_mp_prefixes_multi_origin_handle
- api_mp_routes_handle
- api_vpn_routes_handle

これらのバージョンを使用すると、大きいレポートを作成し、そのレポートを複数に分けて取得した後、レポートを閉じることができます。標準的な方法では、レポートを作成し、すべてのエントリを取得した後、1 回の呼び出しでレポート全体を閉じますが、それとは対照的です。処理に長い時間がかかり、多くのリソースを消費するレポートの場合、レポートを複数の小さい呼び出しに分割することによって、1 つの呼び出しが長時間 RAMS を独占するのを防ぐことができます。

リエントラントクエリーを使用するには、以下の手順を実行します。

1　クエリーのリエントラントバージョンを使用してレポートを作成し、レポートのハンドルを返します。

ハンドルは、後の呼び出しでレポートを指定するために使用する整数です。

これらの呼び出しのリエントラントバージョンで使用するパラメータは、標準バージョンのパラメータと同じです。唯一の違いは、レポート全体の代わりにハンドルが返される点です。

2　api_mp_list_handle 呼び出しを使用して、レポートのうち、ユーザー指定のエントリから始まるユーザー指定の数のエントリを返します。

通常の場合、レポートの最初から同じサイズのまとまりを返し、レポート全体が取得されるまで続けますが、レポートから連続的なまとまりを取得する方法に制限はありません。

3 終了後、api_mp_close_handle を使用してレポートを閉じ、使用されている可能性のあるリソースを解放します。

この処理が終了した後は、api_mp_list_handle を使用してレポートの部分を取得することはできません。

以下の例では、ユーザーが api_mp_events 呼び出しの結果を 1 回に 500 エントリずつ取得できるように、3 種類の呼び出しをすべて使用します。

```
#!/usr/bin/perl

if (!defined($ARGV[0]) || !defined($ARGV[1])) {
   printf "usage: RouteAnalyzer.api_mp_events_handle ip database
   [filter] \n";exit(0);
}

my $rexip = $ARGV[0];
my $database = $ARGV[1];
my $filter = "any";

$filter = $ARGV[2] if ($#ARGV >= 2);

use strict;
use RPC::XML::Client;
use RPC::XML 'time2iso8601';
use Date::Parse;
use Data::Dumper; $Data::Dumper::Terse = 1; $Data::Dumper::
   Indent = 1;
my $client;
my $req;
my @reqs;
my $password = 'admin';
$client = new RPC::XML::Client "http://$rexip:2000/RPC2";

my $t1 = str2time("4 Jul 2000 00:55:17 PST");
my $t2 = str2time("4 Jul 2005 11:55:18 PST");

# 10K entries default; if -1 entered
# RPC implementation will return all
my $num = 10000;
$num = $ARGV[3] if ($#ARGV >= 3);

my $openreq = RPC::XML::request->new(
   RouteAnalyzer.api_mp_events_handle',
   RPC::XML::RPC_STRING($password),
   RPC::XML::RPC_STRING($database),
```

```
   RPC::XML::datetime_iso8601->new(time2iso8601($t1)),
   RPC::XML::datetime_iso8601->new(time2iso8601($t2)),
   RPC::XML::RPC_STRING($filter),
   RPC::XML::RPC_INT($num)
);

my $openres = $client->send_request($openreq);
if ($openres->is_fault) {print("---XMLRPC FAULT ---"); }
my $overall = $openres->value;

my $handle = int($overall->{result});
my $total = int($overall->{numRequestedEntries});

# chunk size is set to 5000
my $step = 5000;

my $index;
my $num;
my $result;
for ($index = 0; $index < $total; $index += $step) {
   my $delta = $total: $index;
   if ($delta > $step) { $delta = $step; }
   my $listreq = RPC::XML::request->new(
      'RouteAnalyzer.api_mp_list_handle',
      RPC::XML::RPC_INT($handle),
      RPC::XML::RPC_STRING($database),
      RPC::XML::RPC_INT($index),
      RPC::XML::RPC_INT($delta),
   );
   my $listresp = $client->send_request($listreq);
   for (@{$listresp->value->{result}}) {
   push @{$result}, $_;
   }
}

my $closereq = RPC::XML::request->new(
   'RouteAnalyzer.api_mp_close_handle',
   RPC::XML::RPC_INT($handle),
   RPC::XML::RPC_STRING($database),
);

my $closeres = $client->send_request($closereq);

$overall->{result} = $result;
my $p = Dumper($overall);
print $p;
```

4　XML-RPC クエリー

本章では、RAMS XML-RPC クエリーの呼び出し、入力パラメータ、結果について説明します。

api_del_watch_list

RPC 呼び出し：Route Analyzer.api_del_watch_list {password} {alert}

このクエリーは、警報に設定済みの監視リストを削除します。監視リストとは、警報の対象となるイベントを限定するために指定された基準のリストであり、基準の形式はプレフィックスとマスクの組み合わせやソースルーターとデスティネーションルーターの組み合わせなど警報により異なります。監視リストを削除すると、警報に対する制限は除去されます。このクエリーは、正常終了の場合は 1 を、それ以外の場合は 0 を返します。

入力パラメータ

- **password**: クエリーに設定されたパスワード。
- **alert**: 監視リストを削除する警報。現在、監視リストを削除できる警報は、AdjLost、AdjEst、RtFlap、LnkFlap、RtOrigChange、RtChange、PeerChange、BgpRouteFlap、BgpLostRedund です。警報の詳細は、『ユーザーガイド』の「警報機能」の章を参照してください。

出力の構造

- ブール

例

```
#!/usr/bin/perl

if(!defined($ARGV[0])) {
   printf "usage:RouteAnalyzer.api_del_watch_list ip\n";
   exit(0);
}

my $rexip = $ARGV[0];

use strict;
use RPC::XML::Client;
use Date::Parse;
use Data::Dumper; $Data::Dumper::Terse = 1; $Data::Dumper::
   Indent = 1;
my $client;
my $req;
my @reqs;
my $password = 'admin';
my $alert = "RtFlap";
$client = new RPC::XML::Client "http://$rexip:2000/RPC2";

push (@reqs,
   RPC::XML::request->new('RouteAnalyzer.api_del_watch_list',
      RPC::XML::RPC_STRING($password),
      RPC::XML::RPC_STRING($alert)
   )
);

foreach (@reqs) {
   my $res = $client->send_request($_);
   if ($res->is_fault) {print("---XMLRPC FAULT ---"); }
   my $value1 = $res->value;
   print (STDERR join "\n", "---XMLRPC RESULT value1---",
      Dumper ($value1) );
}
```

サンプル出力

```
---XMLRPC RESULT value1 ---
'0'
```

api_get_alert_destination_info

RPC 呼び出し：RouteAnalyzer.api_get_alert_destination_info {password} {alert}

このクエリーは、『ユーザーガイド』の「VPN 設定およびレポート」に記述されている Web 管理インタフェースを使用して設定された警報の宛先に関する情報を返します。この情報には、警報の宛先、SNMP トラップのコミュニティ、SNMP トラップの宛先、しきい値、タイムスケールが含まれます。

入力パラメータ

- **password**: クエリーに設定されたパスワード。
- **alert**: 宛先情報を要求する警報。現在、実装されている警告は、AdjLost、AdjEst、RtFlap、LnkFlap、RtOrigChange、RtChange、Event、EChurn、PeerChange、PrefixFlap、BgpPrefixDrought、BgpPrefixFlood、BgpRouteFlap、BgpLostRedund、BgpASPathLonger、BgpDownToOnePath、BgpDownToZeroPaths、BgpAcquiredRedund、RtrReachability、CustReachability、CustPrivacy、RtrIntrusion、BgpPeerLost です。警報の詳細は、『ユーザーガイド』の「Alerts」の章を参照してください。

出力の構造

- vinfo (バージョン情報): バージョン構造体
- alert_dest (警報の宛先):
  - snmp_trap_community (SNMP トラップのコミュニティ): 文字列
  - time_scale (タイムスケール): 整数
  - threshold (しきい値): 整数 (パーセント)
  - alert_destination (警報の宛先): 文字列
  - snmp_trap_destination (SNMP トラップの宛先): 文字列

例

```
#!/usr/bin/perl

if(!defined($ARGV[0])) {
   printf "usage:RouteAnalyzer.api_get_alert_destination_info
   ip\n"; exit(0);
}

my $rexip = $ARGV[0];

use strict;
use RPC::XML::Client;
use Date::Parse;
use Data::Dumper; $Data::Dumper::Terse = 1; $Data::Dumper::
   Indent = 1;
my $client;
my $req;
my @reqs;
my $password = 'admin';
my $alert = "RtFlap";
$client = new RPC::XML::Client "http://$rexip:2000/RPC2";

push (@reqs,
   RPC::XML::request->new('RouteAnalyzer.api_get_alert_
      destination_info',
      RPC::XML::RPC_STRING($password),
      RPC::XML::RPC_STRING($alert)
   )
);

foreach (@reqs) {
   my $res = $client->send_request($_);
   if ($res->is_fault) {print("---XMLRPC FAULT ---"); }
   my $value1 = $res->value;
   print (STDERR join "\n", "---XMLRPC RESULT value1---",
      Dumper ($value1) );
}
```

サンプル出力

```
---XMLRPC RESULT value1 ---
{
   'vinfo' => {
      'software_version' => 'Build 4.0.90 HP RAMS',
      'appliance_version' => '0.5.24'
   },
   'alert_dest' => {
      'snmp_trap_community' => 'public',
      'time_scale' => '60',
      'threshold' => '50',
      'alert_destination' => 'SNMP Trap',
      'snmp_trap_destination' => '192.168.0.123'
   }
}
```

api_get_watch_list

RPC 呼び出し：RouteAnalyzer.api_get_watch_list {password} {alert}

このクエリーは、指定された警報に設定されている監視リストを返します。監視リストとは、警報の対象となるイベントを限定するために指定された基準のリストであり、基準の形式はプレフィックスとマスクの組み合わせやソースルーターとデスティネーションルーターの組み合わせなど警報により異なります。RtFlap 警報に設定されている監視リストをサンプル出力に示します。ソースルーター、デスティネーションルーター、および「op」値が出力されます。「op」値は、「and」、「or」、「none」です。

入力パラメータ

- **password**: クエリーに設定されたパスワード。
- **alert**: 設定されている監視リストを要求する警報。現在、監視リストを取得できる警報は、AdjLost、AdjEst、RtFlap、LnkFlap、RtOrigChange、RtChange、PeerChange、BgpRouteFlap、BgpLostRedund です。警報の詳細は、『ユーザーガイド』の「警報機能」の章を参照してください。

出力の構造

- vinfo (バージョン情報): バージョン構造体
- wlists (監視リスト): 以下の情報の配列
  - src (ソース): IP 構造体
  - dst (デスティネーション): IP 構造体
  - op: 文字列

例

```
#!/usr/bin/perl

if(!defined($ARGV[0])) {
   printf "usage:RouteAnalyzer.api_get_watch_list ip\n";
   exit(0);
}

my $rexip = $ARGV[0];
use strict;
use RPC::XML::Client;
use Date::Parse;
use Data::Dumper; $Data::Dumper::Terse = 1; $Data::Dumper::
   Indent = 1;
my $client;
my $req;
my @reqs;
my $password = 'admin';
my $alert = "RtFlap";
$client = new RPC::XML::Client "http://$rexip:2000/RPC2";

push (@reqs,
   RPC::XML::request->new('RouteAnalyzer.api_get_watch_list',
      RPC::XML::RPC_STRING($password),
      RPC::XML::RPC_STRING($alert)
   )
);

foreach (@reqs) {
   my $res = $client->send_request($_);
   if ($res->is_fault) {print("---XMLRPC FAULT ---"); }
   my $value1 = $res->value;
   print (STDERR join "\n", "---XMLRPC RESULT value1---",
      Dumper ($value1) );
}
```

サンプル出力

```
---XMLRPC RESULT value1 ---
{
   'vinfo' => {
      'software_version' => 'Build 4.0.90 HP RAMS',
      'appliance_version' => '0.5.24'
   },
   'wlists' => [
      {
         'src' => {
         'ip4_addr' => '192.168.0.123'
         },
         'dst' => {
         'ip4_addr' => '192.168.0.40'
         },
         'op' => 'and'
      }
   ]
}
```

api_link_list

RPC 呼び出し：RouteAnalyzer.api_link_list {password} {database name} {time}

このクエリーは、指定されたネットワークのリンクリストを返します。リンクごとに、ソースノードとデスティネーションノードの記述、および両ノード間のインタフェースが出力されます。

> **注記** このクエリーは、将来のリリースで廃止される予定のため推奨できません。新しいアプリケーションには、api_mp_links を使用してください。

入力パラメータ

- **password**: クエリーに設定されたパスワード。
- **database name**: データベース階層に含まれる 1 つ以上のデータベース名 (複数の場合はスペースで区切る)。名前は、下にサブツリーを含む管理ドメイン名 (CorpNet など) か、データベース名全体 (CorpNet.EIGRP/AS100 など) です。
- **time**: UTC タイムゾーンの時刻を ISO 8601 形式で指定します (`20050725T21:47:35` など)。クエリーの結果は、指定された時刻のネットワークの状態に基づいて計算されます。

出力の構造

- vinfo (バージョン情報): バージョン構造体
- network_name (ネットワーク名): 文字列
- report_time (レポート作成時刻): ISO 8601 形式の UTC 時刻
- links (リンク): リンク構造体の配列

例

```
#!/usr/bin/perl

if (!defined($ARGV[0]) || !defined($ARGV[1])) {
   printf "usage:RouteAnalyzer.api_link_list ip database\n";
   exit(0);
}

my $rexip = $ARGV[0];
my $database = $ARGV[1];

use strict;
use RPC::XML::Client;
use RPC::XML 'time2iso8601';
use Date::Parse;
use Data::Dumper; $Data::Dumper::Terse = 1; $Data::Dumper::
   Indent = 1;
my $client;
my $req;
my @reqs;
my $password = 'admin';
$client = new RPC::XML::Client "http://$rexip:2000/RPC2";

my $t1 = str2time("24 Jul 2003 11:53:33 PST");

push (@reqs,
   RPC::XML::request->new('RouteAnalyzer.api_link_list',
      RPC::XML::RPC_STRING($password),
      RPC::XML::RPC_STRING($database),
      RPC::XML::datetime_iso8601->new(time2iso8601($t1))
   )
);

foreach (@reqs) {
   my $res = $client->send_request($_);
   if ($res->is_fault) {print("---XMLRPC FAULT ---"); }
   my $value1 = $res->value;
   print (STDERR join "\n", "---XMLRPC RESULT value1---",
      Dumper ($value1) );
}
```

サンプル出力

```
---XMLRPC RESULT value1 ---
{
   'vinfo' => {
      'software_version' => 'Build 4.0.90 HP RAMS',
      'appliance_version' => '0.5.24'
   },
   'network_name' => 'UCBJul03a',
   'report_time' => '20051027T18:46:09',
   'links' => [
      {
         'source' => {'nodeType' => 'Internal',
            'ip_addr' => {'ip4_addr' => '169.229.128.130'},
            'nodeState' => 'UP',
            'nodeProto' => 'OSPF',
            'name' => '',
            'nodeArea' => 'UCBJul03a.OSPF/169,229,128,128',
            'maskLen' => '32',
            'systemID' => '169,229,128,130'
         },
         'interfaces' => [
            {
               'source' => {'ip4_addr' => '169.229.128.130'},
               'bw' => '0',
               'destination' => {'ip4_addr' => '169.229.128.129'},
               'metric' => '1000',
               'delay' => '10000',
               'state' => '1'
            }
         ],
      'destination' => {
            'nodeType' => 'PseudoNode',
            'ip_addr' => {'ip4_addr' => '169,229,128,129'},
            'nodeState' => 'UP',
            'nodeProto' => 'OSPF',
            'name' => '',
            'nodeArea' => 'UCBJul03a.OSPF/169,229,128,128',
            'maskLen' => '29',
            'systemID' => '169.229.128.129 DR'
         }
      },...
   ]
}
```

api_list_a_route

RPC 呼び出し ：Route Analyzer.api_list_a_route {password} {database name} {source address} {dest prefix} {time}

このクエリーは、ソースからデスティネーションへのパスに関して、要求時のメトリック総数 (計算可能な場合) とリンクリストを返します。同一コストのパスが複数存在する場合でも、1 つのパスのリストのみが返されます。リンクのリストは順に出力されます。

入力パラメータ

- **password**: クエリーに設定されたパスワード。
- **database name**: データベース階層に含まれる 1 つ以上のデータベース名 (複数の場合はスペースで区切る)。名前は、下にサブツリーを含む管理ドメイン名 (CorpNet など) か、データベース名全体 (CorpNet.EIGRP/AS100 など) です。
- **source address**: この XML 構造体には、ルーター ID またはルーターのインタフェースアドレスを示す IPv4 アドレスと、マスク長 32 を指定します。このマスクアドレスは、下位互換のために指定しますが、このクエリーでは無視されます。
- **dest prefix**: この XML 構造体には、IPv4 アドレス (`192.168.123.125` など) とマスク長 (`27` など) で構成されたデスティネーションプレフィックスを指定します。
- **time**: UTC タイムゾーンの時刻を ISO 8601 形式で指定します (`20050725T21:47:35` など)。クエリーの結果は、指定された時刻のネットワークの状態に基づいて計算されます。

出力の構造

- vinfo (バージョン情報): バージョン構造体
- network_name (ネットワーク名): 文字列
- report_time (レポート作成時刻): ISO 8601 形式の UTC 時刻
- route (経路):
  - route_cost (経路コスト): 整数
  - time (時刻): ISO 8601 形式の UTC 時刻
  - links (リンク): リンク構造体の配列

例

```
#!/usr/bin/perl

my $rexip = $ARGV[0];
my $database = $ARGV[1];
use strict;
use RPC::XML::Client;
use RPC::XML 'time2iso8601';
use Date::Parse;
use Data::Dumper; $Data::Dumper::Terse = 1; $Data::Dumper::
   Indent = 1;
my $client; my $req; my @reqs;
my $password = 'admin';

$client = new RPC::XML::Client "http://$rexip:2000/RPC2";

my $t1 = 1099528204;    #seconds since 1970-01-01 00:00:00 UTC

push (@reqs,
   RPC::XML::request->new('RouteAnalyzer.api_list_a_route',
      RPC::XML::RPC_STRING($password),
      RPC::XML::RPC_STRING($database),
      ##### source #####
      new RPC::XML::struct(ip_addr =>
         new RPC::XML::struct(ip4_addr => "192.168.99.99"),
         masklen => 32),
      ##### destination #####
      new RPC::XML::struct(ip_addr =>
         new RPC::XML::struct(ip4_addr => "10.23.113.0"),
         masklen => 27),
      RPC::XML::datetime_iso8601->new(time2iso8601($t1))
   )
);

foreach (@reqs) {
   my $res = $client->send_request($_);
   if ($res->is_fault) {print("---XMLRPC FAULT ---"); }
   my $value1 = $res->value;
   print (join "\n", Dumper($value1));
}
```

サンプル出力

```
{
   'vinfo' => {
      'software_version' => 'Build 4.0.90 HP RAMS',
      'appliance_version' => '0.5.24'
   },
   'network_name' => 'tester',
   'report_time' => '20041104T00:31:49',
   'route' => {
      'route_cost' => '5',
      'time' => '20041104T00:30:04',
      'links' => [
           {'source' => {'nodeType' => 'AreaBR_ASBR',
              'ip_addr' => {'ip4_addr' => '192.168.99.99'},
              'name' => '',
              'nodeArea' => 'tester.OSPF/Backbone',
              'systemID' => '192.168.99.99',
              'nodeState' => 'UP',
              'nodeProto' => 'Unknown',
              'maskLen' => '32'
           },
           'interfaces' => [
              {
                 'source' => {'ip4_addr' => '192.168.116.11'},
                 'bw' => '0',
                 'destination' => {'ip4_addr' => '192.168.116.17'},
                 'metric' => '1',
                 'delay' => '10000',
                 'state' => '1'
              }
           ],
           'destination' => {'nodeType' => 'PseudoNode',
              'ip_addr' => {'ip4_addr' => '192.168.116.17'},
              'name' => '',
              'nodeArea' => 'tester.OSPF/Backbone',
              'systemID' => '192.168.116.17 DR',
              'nodeState' => 'UP',
              'nodeProto' => 'Unknown',
              'maskLen' => '24'
           }
         },...
      ]
   }
}
```

api_list_a_route_ECMP

RPC 呼び出し：RouteAnalyzer.api_list_a_route_ECMP {password} {database name} {source address} {dest prefix} {time}

このクエリーは、ソースからデスティネーションへの 1 つ以上の同一コストパスに関して、要求時のメトリック総数 (計算可能な場合) と同一コストパスを構成するリンクのリストを返します。このリンクリストは、パスセグメントのツリーを横型検索した結果です。そのため、いずれのパスについても、リンクが順に出力されるとは限らず、複数のパスに存在するリンクはリスト内で複数回出力される可能性があります。

入力パラメータ

- **password**: クエリーに設定されたパスワード。
- **database name**: データベース階層に含まれる 1 つ以上のデータベース名 (複数の場合はスペースで区切る)。名前は、下にサブツリーを含む管理ドメイン名 (CorpNet など) か、データベース名全体 (CorpNet.EIGRP/AS100 など) です。
- **source address**: この XML 構造体には、ルーター ID またはルーターのインタフェースアドレスを示す IPv4 アドレスと、マスク長 32 を指定します。このマスクアドレスは、下位互換のために指定しますが、このクエリーでは無視されます。
- **dest prefix**: この XML 構造体には、IPv4 アドレス (`192.168.123.125` など) とマスク長 (`27` など) で構成されたデスティネーションプレフィックスを指定します。
- **time**: UTC タイムゾーンの時刻を ISO 8601 形式で指定します (`20050725T21:47:35` など)。クエリーの結果は、指定された時刻のネットワークの状態に基づいて計算されます。

出力の構造

- vinfo (バージョン情報): バージョン構造体
- network_name (ネットワーク名): 文字列
- report_time (レポート作成時刻): ISO 8601 形式の UTC 時刻
- route (経路):

- route_cost (経路コスト): 整数
- time (時刻): ISO 8601 形式の UTC 時刻
- links (リンク): リンク構造体の配列

例

```
my $rexip = $ARGV[0];
my $database = $ARGV[1];
use strict;
use RPC::XML::Client;
use RPC::XML 'time2iso8601';
use Date::Parse;
use Data::Dumper; $Data::Dumper::Terse = 1; $Data::Dumper::
   Indent = 1;
my $client; my $req; my @reqs;
my $password = 'admin';
$client = new RPC::XML::Client "http://$rexip:2000/RPC2";

my $t1 = str2time("3 Nov 2004 16:30:04 PST");

push (@reqs,
   RPC::XML::request->new('RouteAnalyzer.api_list_a_route_ECMP',
      RPC::XML::RPC_STRING($password),
      RPC::XML::RPC_STRING($database),
      ##### source #####
      new RPC::XML::struct(ip_addr =>
         new RPC::XML::struct(ip4_addr => "192.168.99.99"),
         masklen => 32),
      ##### destination #####
      new RPC::XML::struct(ip_addr =>
         new RPC::XML::struct(ip4_addr => "10.23.113.0"),
         masklen => 27),
      RPC::XML::datetime_iso8601->new(time2iso8601($t1))
   )
);

foreach (@reqs) {
   my $res = $client->send_request($_);
   if ($res->is_fault) {print("---XMLRPC FAULT ---"); }
   my $value1 = $res->value;
   print (join "\n", Dumper($value1));
}
```

サンプル出力

```
{
   'vinfo' => {
      'software_version' => 'Build 4.0.90 HP RAMS',
      'appliance_version' => '0.5.24'
   },
   'network_name' => 'tester',
   'report_time' => '20041105T21:24:16',
   'route' => {
      'route_cost' => '8',
      'time' => '20041104T00:30:04',
      'links' => [
           {'source' => {'nodeType' => 'AreaBR_ASBR',
              'ip_addr' => {'ip4_addr' => '192.168.99.99'},
              'name' => '',
              'nodeArea' => 'tester.OSPF/Backbone',
              'systemID' => '192.168.99.99',
              'nodeState' => 'UP',
              'nodeProto' => 'Unknown',
              'maskLen' => '32'
           },
           'interfaces' => [
              {
                 'source' => {'ip4_addr' => '192.168.99.99'},
                 'bw' => '0',
                 'destination' => {'ip4_addr' => '192.168.107.11'},
                 'metric' => '2',
                 'delay' => '10000',
                 'state' => '1'
              }
           ],
           'destination' => {'nodeType' => 'PseudoNode',
              'ip_addr' => {'ip4_addr' => '192.168.107.11'},
              'name' => '',
              'nodeArea' => 'tester.OSPF/Backbone',
              'systemID' => '192.168.107.11 DR',
              'nodeState' => 'UP',
              'nodeProto' => 'Unknown',
              'maskLen' => '24'
           }
         },...
      ]
   }
}
```

api_mp_close_handle

RPC 呼び出し：RouteAnalyzer.api_mp_close_handle {handle} {database}

このクエリーは、生成済みのレポートハンドルを取得し、レポートを閉じて、レポートが使用しているメモリとリソースを解放します。この後でレポートハンドルを RouteAnalyzer.api_mp_list_handle で使用することはできません。

入力パラメータ

- handle: 「_handle」で終わる RPC 呼び出しで生成した整数。
- database : データベース階層に含まれる 1 つ以上のデータベース名 (複数の場合はスペースで区切る)。名前は、下にサブツリーを含む管理ドメイン名 (CorpNet など) か、データベース名全体 (CorpNet.EIGRP/AS100 など) です。

出力の構造

- vinfo (バージョン情報): バージョン構造体
- numReturned Entries (返すエントリの数): 整数
- network_name (ネットワーク名): 文字列
- report_time (レポート作成時刻): ISO 8601 形式の UTC 時刻
- totalEntries (エントリの総数): 整数
- result (結果): 空白文字列

例 / サンプル

例とサンプル出力の詳細は、27 ページの「リエントラントクエリーの使用方法」を参照してください。

api_mp_events

RPC 呼び出し：RouteAnalyzer.api_mp_events {password} {database name} {time t1} {time t2} {filter} {max entries}

このクエリーは、時刻 t1 から t2 までに発生したマルチプロトコルネットワークイベントのうち、指定されたフィルター条件に一致するイベントのリストを返します。イベントには、BGP プレフィックスのアナウンスや取り消し、IGP 隣接関係の追加や喪失などがあります。

> **注記** このクエリーは、短時間の間に多数の BGP イベントを返す可能性があります。イベント数を管理可能な範囲に抑えるには、フィルターを調節したり、時間を短縮したりします。さらに、このクエリーが時間 t1 から t2 までの BGP イベントをすべて取得するのに必要な時間を確保するために、XML-RPC クライアントのタイムアウトを延長することも必要です。あるいは、オプションパラメータ {max entries} を指定して、返すエントリの数を制限する方法もあります。

入力パラメータ

- **password**: クエリーに設定されたパスワード。
- **database name**: データベース階層に含まれる 1 つ以上のデータベース名 (複数の場合はスペースで区切る)。名前は、下にサブツリーを含む管理ドメイン名 (CorpNet など) か、データベース名全体 (CorpNet.EIGRP/AS100 など) です。
- **time t1、t2**: UTC タイムゾーンの時刻を ISO 8601 形式で指定します (`20050725T21:47:35` など)。クエリーの結果として、ここで指定した 2 つの時刻の間に発生したイベントが出力されます。
- **filter**: フィルターパラメータに一致するサブセットのみを出力するためのフィルター式。フィルター式の詳細は、『ユーザーガイド』の「History Navigator」の章の「式の構文」を参照してください。フィルター「any」を使用すると、結果がすべて返されます。
- **max entries**: クエリーで返すエントリの最大数を示すオプションパラメータであり、32 ビットの整数で指定します。

出力の構造

- vinfo (バージョン情報): バージョン構造体
- numReturnedEntries (返されたエントリの数): 整数
- network_name (ネットワーク名): 文字列
- report_time (レポート作成時刻): ISO 8601 形式の UTC 時刻
- totalEntries (エントリの総数): 整数
- result (結果): 以下の構造体の配列
  - target (ターゲット): 文字列
  - attributesText (属性文字列): 文字列 (BGP でない場合)
  - time (時刻):
    - seconds (秒): 整数
    - useconds (マイクロ秒): 整数
  - topology (トポロジ): トポロジ構造体
  - operation (処理): 文字列
  - router (ルーター): 文字列

例

```
#!/usr/bin/perl

if (!defined($ARGV[0]) || !defined($ARGV[1])) {
   printf "usage:RouteAnalyzer.api_mp_events ip database\n";
   exit(0);
}

my $rexip = $ARGV[0];
my $database = $ARGV[1];
my $filter = "any";
$filter = $ARGV[2] if ($#ARGV >= 2);

use strict;
use RPC::XML::Client;
use RPC::XML 'time2iso8601';
use Date::Parse;
use Data::Dumper; $Data::Dumper::Terse = 1; $Data::Dumper::
   Indent = 1;
my $client;
my $req;
my @reqs;
my $password = 'admin';
$client = new RPC::XML::Client "http://$rexip:2000/RPC2";

my $t1 = str2time("05 Nov 2004 11:09:04 PST");
my $t2 = str2time("05 Nov 2004 11:53:52 PST");

push (@reqs,
   RPC::XML::request->new('RouteAnalyzer.api_mp_events',
      RPC::XML::RPC_STRING($password),
      RPC::XML::RPC_STRING($database),
      RPC::XML::datetime_iso8601->new(time2iso8601($t1)),
      RPC::XML::datetime_iso8601->new(time2iso8601($t2)),
      RPC::XML::RPC_STRING($filter), 150
   )
);

foreach (@reqs) {
   my $res = $client->send_request($_);
   if ($res->is_fault) {print("---XMLRPC FAULT ---"); }
   my $value1 = $res->value;
   print Dumper($value1);
}
```

サンプル出力

```
{
   'vinfo' => {
      'software_version' => 'Build 4.0.90 HP RAMS',
      'appliance_version' => '0.5.24'
   },
   'numReturnedEntries' => '150',
   'network_name' => 'baklab701',
   'report_time' => '20051107T23:13:37:00',
   'totalEntries' => '93971',
   'result' => [
      {
         'target' => '',
         'attributesText' => 'Type:Internal Router',
         'time' => {
            'seconds' => '1130545402',
            'useconds' => '966898'
         },
         'topology' => {
            'fullName' => 'baklab701.OSPF/0.0.0.1',
            'protocol' => 'OSPF'
         },
         'operation' => 'Drop Router',
         'router' => '192.168.0.87'
      },
      {
         'target' => '192.168.0.87',
         'attributesText' => 'Metric:Down',
         'time' => {
            'seconds' => '1130545402',
            'useconds' => '966898'
         },
         'topology' => {
            'fullName' => 'baklab701.OSPF/0.0.0.1',
            'protocol' => 'OSPF'
         },
         'operation' => 'Drop Neighbor',
         'router' => '192.168.0.2 DR'
      },...
   ]
}
```

api_mp_events_handle

RPC 呼び出し：RouteAnalyzer.api_mp_events {password} {database name} {time t1} {time t2} {filter} {max entries}

このクエリーは、時刻 t1 から t2 までに発生したマルチプロトコルネットワークイベントのうち、指定されたフィルター条件に一致するイベントのリストのハンドルを返します。イベントには、BGP プレフィックスのアナウンスや取り消し、IGP 隣接関係の追加や喪失などがあります。

> **注記** このクエリーは、短時間の間に多数の BGP イベントを返す可能性があります。イベント数を管理可能な範囲に抑えるには、フィルターを調節したり、時間を短縮したりします。さらに、このクエリーが時間 t1 から t2 までの BGP イベントをすべて取得するのに必要な時間を確保するために、XML-RPC クライアントのタイムアウトを延長することも必要です。あるいは、オプションパラメータ {max entries} を指定して、返すエントリの数を制限する方法もあります。

入力パラメータ

52 ページの「api_mp_events」の「入力パラメータ」を参照してください。

出力の構造

- vinfo (バージョン情報): バージョン構造体
- numReturnedEntries (返されたエントリの数): 整数
- network_name (ネットワーク名): 文字列
- report_time (レポート作成時刻): ISO 8601 形式の UTC 時刻
- totalEntries (エントリの総数): 整数
- result (結果): 整数

例 / サンプル

例とサンプルの詳細は、27 ページの「リエントラントクエリーの使用方法」を参照してください。

api_mp_osi_routes

RPC 呼び出し：RouteAnalyzer.api_mp_osi_routes {password} {database name} {time} {filter} {max entries}

このクエリーは、Prefix Neighbor と ES Neighbor をアナウンスするすべてのルーターからの経路 (Prefix Neighbor および ES Neighbor アナウンスをすべて含む) のうち、指定された時刻と条件に一致する経路のリストを返します。

> **注記** このクエリーは、短時間の間に多数の経路を返す可能性があります。経路数を管理可能な範囲に抑えるには、フィルターを調節します。さらに、このクエリーが経路をすべて取得するのに必要な時間を確保するために、XML-RPC クライアントのタイムアウトを延長することも必要です。あるいは、オプションパラメータ {max entries} を指定して、返すエントリの数を制限する方法もあります。

入力パラメータ

- **password**: クエリーに設定されたパスワード。
- **database name**: データベース階層に含まれる 1 つ以上のデータベース名 (複数の場合はスペースで区切る)。名前は、下にサブツリーを含む管理ドメイン名 (CorpNet など) か、データベース名全体 (CorpNet.EIGRP/AS100 など) です。
- **time**: UTC タイムゾーンの時刻を ISO 8601 形式で指定します (`20050725T21:47:35` など)。クエリーの結果は、指定された時刻のネットワークの状態に基づいて計算されます。
- **filter**: フィルターパラメータに一致するサブセットのみを出力するためのフィルター式。フィルター式の詳細は、『ユーザーガイド』の「History Navigator」の章の「式の構文」を参照してください。フィルター「any」を使用すると、結果がすべて返されます。
- **max entries**: クエリーで返すエントリの最大数を示すオプションパラメータであり、32 ビットの整数で指定します。

出力の構造

- vinfo (バージョン情報): バージョン構造体
- numReturnedEntries (返されたエントリの数): 整数
- network_name (ネットワーク名): 文字列
- report_time (レポート作成時刻): ISO 8601 形式の UTC 時刻
- totalEntries (エントリの総数): 整数
- result (結果): 以下の構造体の配列
  - topology (トポロジ): トポロジ構造体
  - attributes (属性): ISIS 属性構造体
  - Prefix Neighbor/ES Neighbor: 文字列
  - router (ルーター): 文字列
  - state (状態): MP 状態構造体 (基準値あり)

例

```
#!/usr/bin/perl

if (!defined($ARGV[0]) || !defined($ARGV[1])) {
   printf "usage: RouteAnalyzer.api_mp_osi_routes ip database
      [filter] \n";
   exit(0);
}

my $rexip = $ARGV[0];
my $database = $ARGV[1];
my $filter = "any";

$filter = $ARGV[2] if ($#ARGV >= 2);

use strict;
use RPC::XML::Client;
use RPC::XML 'time2iso8601';
use Date::Parse;
use Data::Dumper; $Data::Dumper::Terse = 1; $Data::Dumper::
   Indent = 1;
my $client;
my $req;
my @reqs;
my $password = 'admin';
$client = new RPC::XML::Client "http://$rexip:2000/RPC2";

my $t1 = str2time("1 Feb 2007 16:50:00 PST");
# 10K entries default
# if -1 entered, RPC implementation will return all
my $num = 2;
$num = $ARGV[3] if ($#ARGV >= 3);

push (@reqs,
   RPC::XML::request->new('RouteAnalyzer.api_mp_osi_routes',
      RPC::XML::RPC_STRING($password),
      RPC::XML::RPC_STRING($database),
      RPC::XML::datetime_iso8601->new(time2iso8601($t1)),
      RPC::XML::RPC_STRING($filter),
      RPC::XML::RPC_INT($num)
   )
);

foreach (@reqs) {
```

```
  my $res = $client->send_request($_);
  if ($res->is_fault) {print("---XMLRPC FAULT ---"); }
  my $value1 = $res->value;
  print Dumper($value1);
}
```

サンプル出力

```
{
   'vinfo' => {
      'software_version' => 'Build 4.0.90 HP RAMS',
      'appliance_version' => '0.5.24'
   },
   'numReturnedEntries' => '2',
   'network_name' => 'PDIHari',
   'report_time' => '20070206T09:03:48:00',
   'totalEntries' => '66',
   'result' => [
      {
         'Prefix Neighbor' => '47.0010.0001',
         'topology' => {
            'fullName' => 'PDIHari.ISIS/Level2',
            'protocol' => 'ISIS'
         },
         'attributes' => {
            'metric' => '0',
            'metricType' => 'Prefix Neighbor Comparable'
         },
         'router' => {
            'sysid' => '47.0024.0000.0001.0000.00,27.0001.0000.
               0001.0000.00',
            'name' => 'Router1',
            'type' => 'L1L2 Router',
            'protoType' => 'OSI'
         },
         'state' => {
            'inBaseline' => 'false',
            'down' => 'true'
         },
         {
            'Prefix Neighbor' => '47.0010.0001',
            'topology' => {
               'fullName' => 'PDIHari.ISIS/Level2',
               'protocol' => 'ISIS'
            },
            'attributes' => {
               'metric' => '0',
               'metricType' => 'Prefix Neighbor Comparable'
            },
            'router' => {
               'sysid' => '47.0023.0000.0021.0000.00,47.0024.0000.
```

```
                0021.0000.00,27.0001.0000.0021.0000.00',
             'name' => 'Router21',
             'type' => 'L1L2 Router',
             'protoType' => 'OSI'
          },
          'state' => {
             'inBaseline' => 'false',
             'down' => 'false'
          }
       }
    }
  ]
}
```

api_mp_osi_routes_handle

RPC 呼び出し：RouteAnalyzer.api_mp_osi_routes {password} {database name} {time} {filter} {max entries}

このクエリーは、Prefix Neighbor と ES Neighbor をアナウンスするすべてのルーターからの経路 (Prefix Neighbor および ES Neighbor アナウンスをすべて含む) のうち、指定された時刻と条件に一致する経路のリストのハンドルを返します。

> **注記** このクエリーは、短時間の間に多数の経路を返す可能性があります。経路数を管理可能な範囲に抑えるには、フィルターを調節します。さらに、このクエリーが経路をすべて取得するのに必要な時間を確保するために、XML-RPC クライアントのタイムアウトを延長することも必要です。あるいは、オプションパラメータ {max entries} を指定して、返すエントリの数を制限する方法もあります。

入力パラメータ

入力パラメータについては、57 ページの「api_mp_osi_routes」の「入力パラメータ」を参照してください。

出力の構造

- vinfo (バージョン情報): バージョン構造体
- numReturnedEntries (返されたエントリの数): 整数
- network_name (ネットワーク名): 文字列
- report_time (レポート作成時刻): ISO 8601 形式の UTC 時刻
- totalEntries (エントリの総数): 整数
- result (結果): 整数

例 / サンプル

例とサンプルの詳細は、27 ページの「リエントラントクエリーの使用方法」を参照してください。

api_mp_links

RPC 呼び出し：RouteAnalyzer.api_mp_links {password} {database name} {time} {filter}

このクエリーは、指定した時刻にマルチプロトコルネットワークに存在するネットワークリンクのリストを返します。結果をフィルタリングして、たとえば単一ルーターに接続するリンクのみを出力することができます。出力は、ソースノード、デスティネーションノード、両ノード間のリンクで構成されます。

入力パラメータ

- **password**: クエリーに設定されたパスワード。
- **database name**: データベース階層に含まれる 1 つ以上のデータベース名 (複数の場合はスペースで区切る)。名前は、下にサブツリーを含む管理ドメイン名 (CorpNet など) か、データベース名全体 (CorpNet.EIGRP/AS100 など) です。
- **time**: UTC タイムゾーンの時刻を ISO 8601 形式で指定します (`20050725T21:47:35` など)。クエリーの結果は、指定された時刻のネットワークの状態に基づいて計算されます。
- **filter**: フィルターパラメータに一致するサブセットのみを出力するためのフィルター式。フィルター式の詳細は、『ユーザーガイド』の「History Navigator」の章の「式の構文」を参照してください。フィルター「any」を使用すると、結果がすべて返されます。

出力の構造

- vinfo (バージョン情報): バージョン構造体
- numReturnedEntries (返されたエントリの数): 整数
- network_name (ネットワーク名): 文字列
- report_time (レポート作成時刻): ISO 8601 形式の UTC 時刻
- totalEntries (エントリの総数): 整数
- result (結果): 以下の構造体の配列
  - link (リンク): MP リンク構造体
  - topology (トポロジ): トポロジ構造体

- sif (ソースインタフェース): 文字列 (IGP がある場合)
- dif (デスティネーションインタフェース): 文字列 (IGP がある場合)
- metric (メトリック): 整数 (IGP がある場合)

例

```
#!/usr/bin/perl

if (!defined($ARGV[0]) || !defined($ARGV[1])) {
   printf "usage:RouteAnalyzer.api_mp_links ip database\n";
   exit(0);
}

my $rexip = $ARGV[0];
my $database = $ARGV[1];
my $filter = "any";

$filter = $ARGV[2] if ($#ARGV >= 2);

use strict;
use RPC::XML::Client;
use RPC::XML 'time2iso8601';
use Date::Parse;
use Data::Dumper; $Data::Dumper::Terse = 1; $Data::Dumper::
   Indent = 1;
my $client;
my $req;
my @reqs;
my $password = 'admin';
$client = new RPC::XML::Client "http://$rexip:2000/RPC2";

my $t1 = str2time("05 Nov 2004 02:11:27 PST");

push (@reqs,
   RPC::XML::request->new('RouteAnalyzer.api_mp_links',
      RPC::XML::RPC_STRING($password),
      RPC::XML::RPC_STRING($database),
      RPC::XML::datetime_iso8601->new(time2iso8601($t1)),
      RPC::XML::RPC_STRING($filter)
   )
);

foreach (@reqs) {
   my $res = $client->send_request($_);
   if ($res->is_fault) {print("---XMLRPC FAULT ---"); }
   my $value1 = $res->value;
   print Dumper($value1);
}
```

サンプル出力

```
{
   'vinfo' => {
      'software_version' => 'Build 4.0.90 HP RAMS',
      'appliance_version' => '0.5.24'
   },
   'numReturnedEntries' => '150',
   'network_name' => 'LabRight',
   'report_time' => '20050725T23:40:42',
   'totalEntries' => '150',
   'result' => [
      {
         'link' => {
            'srcNode' => {
               'type' => 'Route Explorer',
               'ipaddr' => '192.168.122.90'
               },
               'dstNode' => {
                  'type' => 'IBGP Peer',
                  'ipaddr' => '192,168,100,100'
               },
               'state' => {
                  'down' => 'false'
               }
            },
            'topology' => {
               'fullName' => 'LabRight.ConfedsTest.ConfedTestTop.BGP
                  /AS65510',
               'protocol' => 'BGP'
            }
         },
         {
         'link' => {
            'srcNode' => {'type' => ..., 'ipaddr' => ...},
            'dstNode' => {'type' => ..., 'ipaddr' => ...},
            'state' => {'down' => ...}, //end of link
            'dif' => ...,
            'sif' => ...,
            'topology' => {'fullName' => ..., 'protocol' => ...}
         }, //end of topology
         'metric' => ...
      }
   ]
}
```

api_mp_list_handle

RPC 呼び出し：RouteAnalyzer.api_mp_list_handle {handle} {database} {index} {delta}

このクエリーは、生成済みのレポートハンドルを取得し、レポートのうち、ユーザー指定のエントリから始まるユーザー指定の数のエントリを返します。

入力パラメータ

- **handle**: 「_handle」で終わる RPC 呼び出しで生成した整数。
- **database**: データベース階層に含まれる 1 つ以上のデータベース名 (複数の場合はスペースで区切る)。名前は、下にサブツリーを含む管理ドメイン名 (CorpNet など) か、データベース名全体 (CorpNet.EIGRP/AS100 など) です。
- **index**: 最初に返すレポート内のエントリ。
- **delta**: 情報を返すエントリの数。

出力の構造

- vinfo (バージョン情報): バージョン構造体
- numReturnedEntries (返されたエントリの数): 整数
- network_name (ネットワーク名): 文字列
- report_time (レポート作成時刻): ISO 8601 形式の UTC 時刻
- totalEntries (エントリの総数): 整数
- result (結果): 呼び出すレポートにより異なります。

api_mp_list_paths

RPC 呼び出し : RouteAnalyzer.api_mp_list_paths {password} {database name} {source address} {dest prefix} {time}

このクエリーは、ソースからデスティネーションへのパスに関して、要求時のメトリック総数 (計算可能な場合) とそのコストを持つすべてのパスのリストを返します。各パスには、そのパスに関する情報とパスの各ホップの説明が含まれます。

入力パラメータ

- **password**: クエリーに設定されたパスワード。
- **database name**: データベース階層に含まれる 1 つ以上のデータベース名 (複数の場合はスペースで区切る)。名前は、下にサブツリーを含む管理ドメイン名 (CorpNet など) か、データベース名全体 (CorpNet.EIGRP/AS100 など) です。
- **source address**: この XML 構造体には、ルーター ID またはルーターのインタフェースアドレスを示す IPv4 アドレスと、マスク長 32 を指定します。このマスクアドレスは、下位互換のために指定しますが、このクエリーでは無視されます。OSI ネットワークの場合、ソースアドレスは、中継システムのシステム ID を示す XML 構造体です。
- **dest prefix**: この XML 構造体には、IPv4 アドレス (`192.168.123.125` など) とマスク長 (`27` など) で構成されたデスティネーションプレフィックスを指定します。OSI ネットワークの場合、デスティネーションプレフィックスは NSAP です。
- **time**: UTC タイムゾーンの時刻を ISO 8601 形式で指定します (`20050725T21:47:35` など)。クエリーの結果は、指定された時刻のネットワークの状態に基づいて計算されます。

出力の構造

- vinfo (バージョン情報): バージョン構造体
- numReturnedEntries (返されたエントリの数): 整数
- network_name (ネットワーク名): 文字列
- report_time (レポート作成時刻): ISO 8601 形式の UTC 時刻
- totalEntries (エントリの総数): 整数

- result (結果): 以下の構造体の配列
  - link (リンク): MP リンク構造体
  - topology (トポロジ): トポロジ構造体
  - path_src (パスのソース): ルーター構造体
  - path_dst (パスのデスティネーション): プレフィックス IP 構造体
  - path_cost (パスコスト): 整数
  - paths (パス): 以下の情報の配列
    - path (パス): 文字列
    - cost (コスト): 整数
    - num_hops (ホップ数): 以下の情報の配列
    - hops_src (ホップのソース): ルーター構造体
    - hop_dst (ホップのデスティネーション): ルーター構造体
    - area/AS (領域 /AS): 文字列 (該当する場合)
    - interfaces (インタフェース): 以下の情報の配列
    - sif (ソースインタフェース): MP IP 構造体またはエラー文字列のいずれか (IGP の場合)。OSI ネットワークの場合は該当しません。
    - dif (デスティネーションインタフェース): MP IP 構造体またはエラー文字列のいずれか (IGP の場合)。OSI ネットワークの場合は該当しません。
    - bw (帯域幅): 整数 (EIGRP の場合。帯域幅の逆数、Bps 単位、2.56 * 10^{12} をかける)
    - delay (遅延): 整数 (EIGRP の場合。10 マイクロ秒 * 256 単位)
    - bw (帯域幅): 整数 (EIGRP の場合。帯域幅の逆数、Bps 単位、2.56 * 10^{12} をかける)
    - delay (遅延): 整数
    - metric (メトリック): 整数 (IGP の場合)
    - protocol (プロトコル): 文字列
    - prefix (プレフィックス): プレフィックス構造体

例

```
#!/usr/bin/perl

my $rexip = $ARGV[0];
my $database = $ARGV[1];
use strict;
use RPC::XML::Client;
use RPC::XML 'time2iso8601';
use Date::Parse;
use Data::Dumper; $Data::Dumper::Terse = 1;$Data::Dumper::Indemy
$password = 'admin';nt = 1;
my $client; my $req; my @reqs;
my $password = 'admin';
$client = new RPC::XML::Client "http://$rexip:2000/RPC2";

my $t1 = 1099528204;    #seconds since 1970-01-01 00:00:00 UTC

push (@reqs,
   RPC::XML::request->new('RouteAnalyzer.api_list_a_route',
      RPC::XML::RPC_STRING($password),
      RPC::XML::RPC_STRING($database),
   ##### source #####
   new RPC::XML::struct(ip_addr =>
      new RPC::XML::struct(ip4_addr => "24.0.0.2"),
      masklen => 32),
   ##### destination #####
   new RPC::XML::struct(ip_addr =>
      new RPC::XML::struct(ip4_addr => "24.0.0.12"),
      masklen => 27),
   RPC::XML::datetime_iso8601->new(time2iso8601($t1))
   )
);

foreach (@reqs) {
   my $res = $client->send_request($_);
   if ($res->is_fault) {print("---XMLRPC FAULT ---"); }
   my $value1 = $res->value;
   print (join "\n", Dumper($value1));
}
```

サンプル出力

```
{
   'vinfo' => {
      'software_version' => 'Build 4.0.90 HP RAMS',
      'appliance_version' => '0.5.24'
   },
   'numReturnedEntries' => '1',
   'network_name' => 'lab',
   'report_time' => '20061103T21:07:42',
   'totalEntries' => '1',
   'result' => [
      {
         'path_dst' => {
            'masklen' => '32',
            'ip_addr' => {'ip4_addr' => '24.0.0.12'}
         },
         'path_cost' => '-1',
         'path_src' => {
            'type' => 'ASBR',
            'ipaddr' => {'ip4_addr' => '24.0.0.2'}
         },
         'paths' => [
            {
               'cost' => '2',
               'path_hops' => [
                  {
                     'hop_src' => {
                        'type' => 'ASBR',
                        'ipaddr' => {'ip4_addr' => '24.0.0.2'}
                     },
                     'protocol' => 'OSPF',
                     'hop_dst' => {
                        'type' => 'LAN Pseudo-Node',
                        'ipaddr' => {'ip4_addr' => '192.168.101.2'},
                        'interfaces' => [
                           {
                              'dif' => 'Not available',
                              'sif' => {'ip4_addr' => '192.168.101.2'}
                           }
                        ],
                        'area/AS' => 'lab.OSPF/0.0.0.1',
                        'metric' => '1',
                        'prefix' => {
                           'masklen' => '32',
```

```
                        'ip_addr' => {'ip4_addr' => '24.0.0.12'}
                    }
                },...
            }
        ],
        'path' => 'Path 1',
        'num_hops' => '3'
    },...
]
}
]
}
```

例 (OSI ネットワーク)

```
#!/usr/bin/perl
#*
#*

if(!defined($ARGV[0]) || !defined($ARGV[1]) || !defined($ARGV[2])
|| !defined($ARGV[3])) {
   printf "usage: RouteAnalyzer.api_mp_list_osi_paths ip database
      src dest\n";
   exit(0);
}

my $rexip = $ARGV[0];
my $database = $ARGV[1];
my $srcaddr = $ARGV[2];
my $dstaddr = $ARGV[3];

use strict;
use RPC::XML::Client;
use RPC::XML 'time2iso8601';
use Date::Parse;
use Data::Dumper; $Data::Dumper::Terse = 1; $Data::Dumper::
   Indent = 1;
my $client;
my $req;
my @reqs;
my $password = 'admin';
$client = new RPC::XML::Client "http://$rexip:2000/RPC2";

my $t1 = str2time("01 Feb 2007 16:40:21 PST");

push (@reqs,
   RPC::XML::request->new('RouteAnalyzer.api_mp_list_paths',
      RPC::XML::RPC_STRING($password),
      RPC::XML::RPC_STRING($database),
      new RPC::XML::struct(ip_addr =>
         new RPC::XML::struct(osi_addr => "$srcaddr"),
         ## srcaddr = 0000.0001.0000.00
         masklen => 0),
         # mask len is not used
      new RPC::XML::struct(ip_addr =>
         new RPC::XML::struct(osi_addr => "$dstaddr"),
         ## dstaddr = 27.0001.0000.0008.0000.00
         masklen => 0),
```

```
        #masklen is not used.
      RPC::XML::datetime_iso8601->new(time2iso8601($t1))
   )
);

foreach (@reqs) {
   my $res = $client->send_request($_);
   if ($res->is_fault) {print("---XMLRPC FAULT ---"); }
   my $value1 = $res->value;
   print Dumper($value1);
}
```

サンプル出力 (OSI ネットワーク)

```
{
   'vinfo' => {
      'software_version' => 'Build 4.0.90 HP RAMS',
      'appliance_version' => '0.5.24'
   },
   'numReturnedEntries' => '1',
   'network_name' => 'PDIHari',
   'report_time' => '20070206T09:15:23:00',
   'totalEntries' => '1',
   'result' => [
      {
         'path_dst' => '27.0001.0000.0008.0000.00',
         'path_cost' => '10',
         'path_src' => {
            'sysid' => '47.0024.0000.0001.0000.00,27.0001.0000.
               0001.0000.00',
            'name' => 'Router1',
            'type' => 'L1L2 Router',
            'protoType' => 'OSI'
         },
         'paths' => [
            {
               'cost' => '10',
               'path_hops' => [
                  {
                     'hop_src' => {
                        'sysid' => '47.0024.0000.0001.0000.00,27.
                           0001.0000.0001.0000.00',
                        'name' => 'Router1',
                        'type' => 'L1L2 Router',
                        'protoType' => 'OSI'
                     },
                     'protocol' => 'ISIS',
                     'hop_dst' => {
                        'sysid' => '47.0023.0000.0021.0000.01,47.
                           0024.0000.0021.0000.01,27.0001.0000.0021.
                           0000.01',
                        'type' => 'LAN Pseudo-Node',
                        'protoType' => 'OSI'
                     },
                     'metric' => '10',
                     'prefix' => '27.0001.0000.0008.0000.00'
                  },
```

```
                    {
                       'hop_src' => {
                          'sysid' => '47.0023.0000.0021.0000.01,47.
                             0024.0000.0021.0000.01,27.0001.0000.0021.
                             0000.01',
                          'type' => 'LAN Pseudo-Node',
                          'protoType' => 'OSI'
                       },
                       'protocol' => 'ISIS',
                       'hop_dst' => {
                          'sysid' => '27.0001.0000.0008.0000.00',
                          'name' => 'Router8',
                          'type' => 'L1 Internal Router',
                          'protoType' => 'OSI'
                       },
                       'metric' => '0',
                       'prefix' => '27.0001.0000.0008.0000.00'
                    },
                    {
                       'hop_src' => {
                          'sysid' => '27.0001.0000.0008.0000.00',
                          'name' => 'Router8',
                          'type' => 'L1 Internal Router',
                          'protoType' => 'OSI'
                       },
                       'protocol' => 'ISIS',
                       'hop_dst' => {
                          'sysid' => '27.0001.0000.0008.0000.00',
                          'name' => 'Router8',
                          'type' => 'L1 Internal Router',
                          'protoType' => 'OSI'
                       },
                       'metric' => '0',
                       'prefix' => '27.0001.0000.0008.0000.00'
                    }
                 ],
                 'path' => 'Path 1',
                 'num_hops' => '3'
              }
           ]
        }
     ]
  }
```

api_mp_prefixes_multi_origin

RPC 呼び出し：RouteAnalyzer.api_mp_prefixes_multi_origin {password} {database name} {time}{threshold} {max entries}

このクエリーは、指定されたネットワークのプレフィックスのうち、複数のルーター (OSI 用語では中継システム) が通知するプレフィックスのリストのハンドルを返します。

入力パラメータ

- **password**: クエリーに設定されたパスワード。
- **database name**: データベース階層に含まれる 1 つ以上のデータベース名 (複数の場合はスペースで区切る)。名前は、下にサブツリーを含む管理ドメイン名 (CorpNet など) か、データベース名全体 (CorpNet.EIGRP/AS100 など) です。
- **time**: UTC タイムゾーンの時刻を ISO 8601 形式で指定します (`20050725T21:47:35` など)。クエリーの結果は、指定された時刻のネットワークの状態に基づいて計算されます。
- **threshold**: レポートに記録するプレフィックスの通知ルーターの最大数を表すしきい値。最大数は 2 です。
- **max entries**: クエリーで返すエントリの最大数を示すオプションパラメータであり、32 ビットの整数で指定します。

出力の構造

- vinfo (バージョン情報): バージョン構造体
- numReturnedEntries (返されたエントリの数): 整数
- network_name (ネットワーク名): 文字列
- report_time (レポート作成時刻): ISO 8601 形式の UTC 時刻
- totalEntries (エントリの総数): 整数
- prefixes (プレフィックス): 以下のプレフィックスの配列
  - prefix (IP プレフィックスまたは Prefix Neighbors/ES Neighbor) : 文字列
  - prefix_area (プレフィックスの領域): 文字列

- prefix_type (プレフィックスのタイプ): 文字列
- router (ルーター): MP ルーター構造体

例

```
#!/usr/bin/perl

if (!defined($ARGV[0]) || !defined($ARGV[1])) {
   printf "usage: RouteAnalyzer.api_mp_prefixes_multi_origin ip
      database [filter] \n";
   exit(0);
}

my $rexip = $ARGV[0];
my $database = $ARGV[1];
my $thresh = 2;

$thresh = $ARGV[2] if ($#ARGV >= 2);

use strict;
use RPC::XML::Client;
use RPC::XML 'time2iso8601';
use Date::Parse;
use Data::Dumper; $Data::Dumper::Terse = 1; $Data::Dumper::
   Indent = 1;
my $client;
my $req;
my @reqs;
my $password = 'admin';
$client = new RPC::XML::Client "http://$rexip:2000/RPC2";

my $t1 = str2time("5 Feb 2007 19:50:00 PST");

# 10K entries default; if -1 entered
# RPC implementation will return all
my $num = 10000;
$num = $ARGV[3] if ($#ARGV >= 3);

push (@reqs,
   RPC::XML::request->new('RouteAnalyzer.api_mp_prefixes_multi_
      origin',
      RPC::XML::RPC_STRING($password),
      RPC::XML::RPC_STRING($database),
      RPC::XML::datetime_iso8601->new(time2iso8601($t1)),
      RPC::XML::RPC_INT($thresh),
      RPC::XML::RPC_INT($num)
   )
);
```

```
foreach (@reqs) {
   my $res = $client->send_request($_);
   if ($res->is_fault) {print("---XMLRPC FAULT ---"); }
   my $value1 = $res->value;
   print Dumper($value1);
}
```

サンプル出力

```
{
   'vinfo' => {
      'software_version' => 'Build 4.0.90 HP RAMS',
      'appliance_version' => '0.5.24'
   },
   'numReturnedEntries' => '6',
   'network_name' => 'PDIHari',
   'report_time' => '20070206T18:33:03',
   'totalEntries' => '6',
   'result' => [
      {
         'nsap' => '47.0010.0001',
         'area' => 'PDIHari.ISIS/Level2',
         'type' => 'Prefix Neighbor',
         'router' => {
            'sysid' => '47.0024.0000.0001.0000.00,27.0001.0000.0001.
               0000.00',
            'name' => 'Router1',
            'type' => 'L1L2 Router',
            'protoType' => 'OSI'
         }
      },
      {
         'nsap' => '47.0010.0001',
         'area' => 'PDIHari.ISIS/Level2',
         'type' => 'Prefix Neighbor',
         'router' => {
            'sysid' => '47.0023.0000.0021.0000.00,47.0024.0000.0021.
               0000.00,27.0001.0000.0021.0000.00',
            'name' => 'Router21',
            'type' => 'L1L2 Router',
            'protoType' => 'OSI'
         }
      },
      {
         'nsap' => '47.0011.0001',
         'area' => 'PDIHari.ISIS/Level2',
         'type' => 'Prefix Neighbor',
         'router' => {
            'sysid' => '47.0024.0000.0001.0000.00,27.0001.0000.
               0001.0000.00',
            'name' => 'Router1',
            'type' => 'L1L2 Router',
```

```
            'protoType' => 'OSI'
          }
        },
        {
          'nsap' => '47.0011.0001',
          'area' => 'PDIHari.ISIS/Level2',
          'type' => 'Prefix Neighbor',
          'router' => {
            'sysid' => '47.0023.0000.0021.0000.00,47.0024.0000.0021.
              0000.00,27.0001.0000.0021.0000.00',
            'name' => 'Router21',
            'type' => 'L1L2 Router',
            'protoType' => 'OSI'
          }
        },
        {
          'nsap' => '00',
          'area' => 'PDIHari.ISIS/Level2',
          'type' => 'Prefix Neighbor',
          'router' => {
            'sysid' => '47.0024.0000.0001.0000.00,27.0001.0000.
              0001.0000.00',
            'name' => 'Router1',
            'type' => 'L1L2 Router',
            'protoType' => 'OSI'
          }
        },
        {
          'nsap' => '00',
          'area' => 'PDIHari.ISIS/Level2',
          'type' => 'Prefix Neighbor',
          'router' => {
            'sysid' =>
            '47.0023.0000.0021.0000.00,47.0024.0000.0021.0000.00,27.
              0001.0000.0021.0000.00',
            'name' => 'Router21',
            'type' => 'L1L2 Router',
            'protoType' => 'OSI'
          }
        }
      ]
    }
```

api_mp_prefixes_multi_origin_handle

RPC 呼び出し：RouteAnalyzer.api_mp_prefixes_multi_origin_handle {password} {database name} {time} {filter} {max entries}

このクエリーは、指定されたネットワークのプレフィックスのうち、複数のルーター (OSI 用語では中継システム) が通知するプレフィックスのリストのハンドルを返します。

入力パラメータ

入力パラメータについては、「api_mp_prefixes_multi_origin」を参照してください。

出力の構造

- vinfo (バージョン情報): バージョン構造体
- numReturnedEntries (返されたエントリの数): 整数
- network_name (ネットワーク名): 文字列
- report_time (レポート作成時刻): ISO 8601 形式の UTC 時刻
- totalEntries (エントリの総数): 整数
- prefixes (プレフィックス): 以下の情報の配列
  - prefix (IP プレフィックスまたは Prefix Neighbors/ES Neighbor) : 文字列
  - prefix_area (プレフィックスの領域): 文字列
  - prefix_type (プレフィックスのタイプ): 文字列
  - router (ルーター): MP ルーター構造体

例 / サンプル

例とサンプル出力の詳細は、27 ページの「リエントラントクエリーの使用方法」を参照してください。

api_mp_routes

RPC 呼び出し：RouteAnalyzer.api_mp_routes {password} {database name} {time} {filter} {max entries}

このクエリーは、プレフィックスをアナウンスするすべてのルーターからの経路 (プレフィックスアナウンスを含む) のうち、指定された時刻とフィルター条件に一致する経路のリストを返します。

> **注記** このクエリーは、短時間の間に多数の BGP 経路を返す可能性があります。経路数を管理可能な範囲に抑えるには、フィルターを調節します。さらに、このクエリーが経路をすべて取得するのに必要な時間を確保するために、XML-RPC クライアントのタイムアウトを延長することも必要です。あるいは、オプションパラメータ {max entries} を指定して、返すエントリの数を制限する方法もあります。

入力パラメータ

- **password**: クエリーに設定されたパスワード。
- **database name**: データベース階層に含まれる 1 つ以上のデータベース名 (複数の場合はスペースで区切る)。名前は、下にサブツリーを含む管理ドメイン名 (CorpNet など) か、データベース名全体 (CorpNet.EIGRP/AS100 など) です。
- **time**: UTC タイムゾーンの時刻を ISO 8601 形式で指定します (`20050725T21:47:35` など)。クエリーの結果は、指定された時刻のネットワークの状態に基づいて計算されます。
- **filter**: フィルターパラメータに一致するサブセットのみを出力するためのフィルター式。フィルター式の詳細は、『ユーザーガイド』の「History Navigator」の章の「式の構文」を参照してください。フィルター「any」を使用すると、結果がすべて返されます。
- **max entries**: クエリーで返すエントリの最大数を示すオプションパラメータであり、32 ビットの整数で指定します。

出力の構造

- vinfo (バージョン情報): バージョン構造体
- numReturnedEntries (返されたエントリの数): 整数

- network_name (ネットワーク名): 文字列
- report_time (レポート作成時刻): ISO 8601 形式の UTC 時刻
- totalEntries (エントリの総数): 整数
- result (結果): 以下の情報の配列
  - topology (トポロジ): トポロジ構造体
  - attributes (属性): LS 属性構造体 (LS の場合)
  - attributes (属性): EIGRP 属性構造体 (EIGRP の場合)
  - attributes (属性): 文字列 (他の IGP の場合)
  - attributes (属性): 静的属性構造体 (静的な経路の場合)
  - attributes (属性): BGP 属性構造体 (BGP の場合)
  - attributes (属性): 文字列 (他のプロトコルの場合)
  - prefix (プレフィックス): 文字列
  - router (ルーター): MP ルーター構造体
  - state (状態): MP 状態構造体 (基準値あり)

例

```
#!/usr/bin/perl

if (!defined($ARGV[0]) || !defined($ARGV[1])) {
   printf "usage: RouteAnalyzer.api_mp_routes ip database
      filter\n";
   exit(0);
}

my $rexip = $ARGV[0];
my $database = $ARGV[1];
my $filter = "any";
$filter = $ARGV[2] if ($#ARGV >= 2);

use strict;
use RPC::XML::Client;
use RPC::XML 'time2iso8601';
use Date::Parse;
use Data::Dumper; $Data::Dumper::Terse = 1; $Data::Dumper::
   Indent = 1;
my $client;
my $req;
my @reqs;
my $password = 'admin';
$client = new RPC::XML::Client "http://$rexip:2000/RPC2";

my $t1 = time;

push (@reqs,
   RPC::XML::request->new('RouteAnalyzer.api_mp_routes',
      RPC::XML::RPC_STRING($password),
      RPC::XML::RPC_STRING($database),
      RPC::XML::datetime_iso8601->new(time2iso8601($t1)
   ),
   RPC::XML::RPC_STRING($filter), 150)
);

foreach (@reqs) {
   my $res = $client->send_request($_);
   if ($res->is_fault) {print("---XMLRPC FAULT ---"); }
   my $value1 = $res->value;
   print Dumper($value1);
}
```

サンプル出力

```
{
   'vinfo' => {
      'software_version' => 'Build 4.0.90 HP RAMS',
      'appliance_version' => '0.5.24'
   },
   'numReturnedEntries' => '150',
   'network_name' => 'baklab701',
   'report_time' => '20051107T23:07:45:00',
   'totalEntries' => '680',
   'result' => [
      {
         'topology' => {
            'fullName' => 'AllStaticRoutes.Static',
            'protocol' => 'Static'
         },
         'attributes' => {
            'nextHops' => [
               {
                  'nextHop' => '192.168.101.101/8'
               }
            ]
         },
         'prefix' => '0.0.0.0/0',
         'router' => {
            'type' => 'Static',
            'ipaddr' => '192.168.133.34'
         },
         'state' => {
            'inBaseline' => 'false',
            'down' => 'false'
         }
      },...
   ]
}
```

api_mp_routes_handle

RPC 呼び出し： RouteAnalyzer.api_mp_routes {password} {database name} {time} {filter} {max entries}

このクエリーは、プレフィックスをアナウンスするすべてのルーターからの経路 (プレフィックスアナウンスを含む) のうち、指定された時刻とフィルター条件に一致する経路のリストのハンドルを返します。

> **注記** このクエリーは、短時間の間に多数の BGP 経路を返す可能性があります。経路数を管理可能な範囲に抑えるには、フィルターを調節します。さらに、このクエリーが経路をすべて取得するのに必要な時間を確保するために、XML-RPC クライアントのタイムアウトを延長することも必要です。あるいは、オプションパラメータ {max entries} を指定して、返すエントリの数を制限する方法もあります。

入力パラメータ

入力パラメータについては、85 ページの「api_mp_routes」を参照してください。

出力の構造

- vinfo (バージョン情報): バージョン構造体
- numReturnedEntries (返されたエントリの数): 整数
- network_name (ネットワーク名): 文字列
- report_time (レポート作成時刻): ISO 8601 形式の UTC 時刻
- totalEntries (エントリの総数): 整数
- result (結果): 整数

例 / サンプル

例とサンプル出力の詳細は、27 ページの「リエントラントクエリーの使用方法」を参照してください。

api_mp_routers

RPC 呼び出し：RouteAnalyzer.api_mp_routers {password} {database name} {time} {filter}

このクエリーは、指定した時刻にマルチプロトコルネットワークに存在するルーターのリストを返します。結果をフィルタリングして、たとえば特定のプロトコルを実行するルーターのみを出力することができます。

入力パラメータ

- **password**: クエリーに設定されたパスワード。
- **database name**: データベース階層に含まれる 1 つ以上のデータベース名 (複数の場合はスペースで区切る)。名前は、下にサブツリーを含む管理ドメイン名 (CorpNet など) か、データベース名全体 (CorpNet.EIGRP/AS100 など) です。
- **time**: UTC タイムゾーンの時刻を ISO 8601 形式で指定します (`20050725T21:47:35` など)。クエリーの結果は、指定された時刻のネットワークの状態に基づいて計算されます。
- **filter**: フィルターパラメータに一致するサブセットのみを出力するためのフィルター式。フィルター式の詳細は、『ユーザーガイド』の「History Navigator」の章の「式の構文」を参照してください。フィルター「any」を使用すると、結果がすべて返されます。

出力の構造

- vinfo (バージョン情報): バージョン構造体
- numReturnedEntries (返されたエントリの数): 整数
- network_name (ネットワーク名): 文字列
- report_time (レポート作成時刻): ISO 8601 形式の UTC 時刻
- totalEntries (エントリの総数): 整数
- result (結果): 以下の情報の配列
  - topology (トポロジ): トポロジ構造体
  - numPrefixes (プレフィックス数): 整数

- router (ルーター): MP ルーター構造体
- state (状態): MP 状態構造体 (基準値なし)

例

```
#!/usr/bin/perl

if (!defined($ARGV[0]) || !defined($ARGV[1])) {
   printf "usage:RouteAnalyzer.api_mp_routers ip database\n";
   exit(0);
}

my $rexip = $ARGV[0];
my $database = $ARGV[1];
my $filter = "any";
$filter = $ARGV[2] if ($#ARGV >= 2);

use strict;
use RPC::XML::Client;
use RPC::XML 'time2iso8601';
use Date::Parse;
use Data::Dumper; $Data::Dumper::Terse = 1; $Data::Dumper::
   Indent = 1;
my $client;
my $req;
my @reqs;
my $password = 'admin';
$client = new RPC::XML::Client "http://$rexip:2000/RPC2";

my $t1 = str2time("28 Feb 2005 15:50:22 PST");

push (@reqs,
   RPC::XML::request->new('RouteAnalyzer.api_mp_routers',
      RPC::XML::RPC_STRING($password),
      RPC::XML::RPC_STRING($database),
      RPC::XML::datetime_iso8601->new(time2iso8601($t1)),
      RPC::XML::RPC_STRING($filter)
   )
);

foreach (@reqs) {
   my $res = $client->send_request($_);
   if ($res->is_fault) {print("---XMLRPC FAULT ---"); }
   my $value1 = $res->value;
   print Dumper($value1);
}
```

サンプル出力

```
{
   'vinfo' => {
      'software_version' => 'Build 4.0.90 HP RAMS',
      'appliance_version' => '0.5.24'
   },
   'numReturnedEntries' => '100',
   'network_name' => 'pd353',
   'report_time' => '20051027T23:33:57',
   'totalEntries' => '100',
   'result' => [
      {
         'topology' => {
            'fullName' => 'pd353.Left.BGP/AS65522',
            'protocol' => 'BGP'
         },
         'numPrefixes' => '0',
         'router' => {
            'type' => 'Route Explorer',
            'ipaddr' => '192.168.0.49'
         },
         'state' => {
            'down' => 'false'
         }
      },
      {
         'topology' => {
            'fullName' => 'pd353.Left.ISIS/Level2',
            'protocol' => 'ISIS'
         },
         'numPrefixes' => '3',
         'router' => {
            'name' => 'labnet-gw',
            'type' => 'AreaBR',
            'ipaddr' => '192,168,104,254'
         },
         'state' => {
            'down' => 'false'
         }
      },...
   ]
}
```

api_prefix_events

RPC 呼び出し : RouteAnalyzer.api_prefix_events {password} {database name} {prefix}

このクエリーは、指定されたプレフィックスのイベントリストを返します。各イベントリストには、プレフィックスおよびプレフィックスを通知したルーターに関する情報が含まれます。

> **注記** このクエリーは、指定されたプレフィックスに関して、要求先データベースが対象とする期間全体のイベントリストを返します。そのため、結果として返されるエントリの数が多くなる可能性があります。エントリ数を制限するには、このクエリーの代わりに api_mp_events クエリーを使用し、フィルターとしてプレフィックスを指定する方法があります。

入力パラメータ

- **password**: クエリーに設定されたパスワード。
- **database name**: データベース階層に含まれる 1 つ以上のデータベース名 (複数の場合はスペースで区切る)。名前は、下にサブツリーを含む管理ドメイン名 (CorpNet など) か、データベース名全体 (CorpNet.EIGRP/AS100 など) です。
- **prefix**: この XML 構造体には、IPv4 アドレス (`192.168.123.125` など) とマスク長 (`27` など) で構成されたプレフィックスを指定します。

出力の構造

- vinfo (バージョン情報): バージョン構造体
- network_name (ネットワーク名): 文字列
- report_time (レポート作成時刻): ISO 8601 形式の UTC 時刻
- events (イベント): 以下の情報の配列
  - prefix_event (プレフィックスのイベント):
    - eventType: 文字列
    - time (時刻): ISO 8601 形式の UTC 時刻
    - usec (マイクロ秒): 整数

- prefix (プレフィックス): プレフィックス構造体
- router (ルーター): ルーター構造体

例

```
#!/usr/bin/perl

if (!defined($ARGV[0]) || !defined($ARGV[1])) {
   printf "usage:RouteAnalyzer.api_prefix_events ip database\n";
   exit(0);
}

my $rexip = $ARGV[0];
my $database = $ARGV[1];

use strict;
use RPC::XML::Client;
use RPC::XML 'time2iso8601';
use Date::Parse;
use Data::Dumper; $Data::Dumper::Terse = 1; $Data::Dumper::
   Indent = 1;
my $client;
my $req;
my @reqs;
my $password = 'admin';
$client = new RPC::XML::Client "http://$rexip:2000/RPC2";

push (@reqs,
   RPC::XML::request->new('RouteAnalyzer.api_prefix_events',
      RPC::XML::RPC_STRING($password),
      RPC::XML::RPC_STRING($database),
      new RPC::XML::struct(ip_addr =>
         new RPC::XML::struct(ip4_addr => "169.229.147.0"),
         masklen => 25)
   )
);

foreach (@reqs) {
   my $res = $client->send_request($_);
   if ($res->is_fault) {print("---XMLRPC FAULT ---"); }
   my $value1 = $res->value;
   print (STDERR join "\n", "---XMLRPC RESULT value1---",
      Dumper ($value1) );
}
```

サンプル出力

```
---XMLRPC RESULT value1 ---
{
   'vinfo' => {
      'software_version' => 'Build 4.0.90 HP RAMS',
      'appliance_version' => '0.5.24'
   },
   'network_name' => 'UCBJul03a',
   'report_time' => '20051027T18:42:29',
   'events' => [
      {
         'prefix_event' => {
            'eventType' => 'advertised',
            'time' => '20030722T22:10:56:00',
            'usec' => '521406',
            'prefix' => {
               'masklen' => '25',
               'ip_addr' => {'ip4_addr' => '169.229.147.0'}
            },
            'router' => {
               'nodeType' => 'AreaBR_ASBR',
               'ip_addr' => {'ip4_addr' => '128.32.1.209'},
               'nodeState' => 'UP',
               'nodeProto' => 'OSPF',
               'name' => '',
               'nodeArea' => 'UCBJul03a.OSPF/169,229,128,128',
               'maskLen' => '32',
               'systemID' => '128.32.1.209'
            }
         }
      },...
   ]
}
```

api_prefix_list

RPC 呼び出し：RouteAnalyzer.api_prefix_list {password} {database name} {time}

このクエリーは、特定の時刻にデータベースで通知されたすべてのプレフィックスのリストを返します。下記のプログラム例は、2002 年 12 月 17 日の午前 0 時に通知されたプレフィックスを要求し、プレフィックスの IP アドレス、タイプ、領域、通知したルーターで構成されるプレフィックス情報を表示します。

> **注記** このクエリーは、将来のリリースで廃止される予定のため推奨できません。新しいアプリケーションには、api_mp_routes を使用してください。

入力パラメータ

- **password**: クエリーに設定されたパスワード。
- **database name**: データベース階層に含まれる 1 つ以上のデータベース名 (複数の場合はスペースで区切る)。名前は、下にサブツリーを含む管理ドメイン名 (CorpNet など) か、データベース名全体 (CorpNet.EIGRP/AS100 など) です。
- **time**: UTC タイムゾーンの時刻を ISO 8601 形式で指定します (`20050725T21:47:35` など)。クエリーの結果は、指定された時刻のネットワークの状態に基づいて計算されます。

出力の構造

- vinfo (バージョン情報): バージョン構造体
- network_name (ネットワーク名): 文字列
- report_time (レポート作成時刻): ISO 8601 形式の UTC 時刻
- prefixes (プレフィックス): 以下の情報の配列
  - prefix_type (プレフィックスのタイプ): 文字列
  - prefix_area (プレフィックスの領域): 文字列
  - prefix (プレフィックス): プレフィックス構造体
  - routers (ルーター): ルーター構造体の配列

例

```
#!/usr/bin/perl

use strict;
use RPC::XML::Client;
use RPC::XML 'time2iso8601';
use Date::Parse;
use Data::Dumper; $Data::Dumper::Terse = 1; $Data::Dumper::
   Indent = 1;

my $t1 = time2iso8601(str2time("17 Dec 2002 00:00:00 PST"));
my $request = RPC::XML::request->new(
   'RouteAnalyzer.api_prefix_list',
   RPC::XML::RPC_STRING( 'packet' ),  //password
   RPC::XML::RPC_STRING( 'CorpNet' ), //database name
   RPC::XML::datetime_iso8601->new($t1)
);

my $client = new RPC::XML::Client 'http://hostname:2000/RPC2';
my $result = $client->send_request($request);
if ($result->is_fault) { print("--- XMLRPC FAULT ---"); }
print(STDERR join"\n","---XMLRPC
RESULT---",Dumper($result->value), '');
```

サンプル出力

```
--- XMLRPC RESULT ---
{
   'vinfo' => {
      'software_version' => 'Build 4.0.90 HP RAMS'
      'appliance_version' => '0.5.24',
   },
   'report_time' => '20030303T20:42:01',
   'network_name' => 'CorpNet',
   'prefixes' => [
      {
         'prefix_type' => 'Internal',
         'prefix_area' => '00000001/ospf',
         'prefix'=>{'ip_addr'=>{'ip4_addr'=>'192.168.240.1'},'maskl
            en'=>'32'},
         'routers' => [
            {
               'nodeProto' => 'ospf',
               'ip_addr' => { 'ip4_addr' => '192.168.104.2' },
               'nodeType' => 'ASBR',
               'name' => '',
               'systemID' => '192168104002:00'
            }
            {'nodeProto'...'ip_addr'...'nodeType'...'name'...
               'systemID'...},
            {'nodeProto'...'ip_addr'...'nodeType'...'name'...
               'systemID'...}
         ] // end routers
      },
      {'prefix_type'...'prefix_area'...'prefix'...'routers'=>
         [...] },
      {'prefix_type'...'prefix_area'...'prefix'...'routers'=>
         [...] }
   ] // end prefixes
}
```

api_prefix_list_filtered

RPC 呼び出し：RouteAnalyzer.api_prefix_list_filtered {password} {database name} {router} {time}

このクエリーは、特定の時刻に指定のルーターによって通知されたプレフィックスのリストを返します。

> **注記** このクエリーは、将来のリリースで廃止される予定のため推奨できません。新しいアプリケーションには、api_mp_routes を使用してください。

入力パラメータ

- **password**: クエリーに設定されたパスワード。
- **database name**: データベース階層に含まれる 1 つ以上のデータベース名 (複数の場合はスペースで区切る)。名前は、下にサブツリーを含む管理ドメイン名 (CorpNet など) か、データベース名全体 (CorpNet.EIGRP/AS100 など) です。
- **router**: この XML 構造体には、ルーター ID またはルーターのインタフェースアドレスを示す IPv4 アドレスと、マスク長 32 を指定します。このマスクアドレスは、下位互換のために指定しますが、このクエリーでは無視されます。
- **time**: UTC タイムゾーンの時刻を ISO 8601 形式で指定します (`20050725T21:47:35` など)。クエリーの結果は、指定された時刻のネットワークの状態に基づいて計算されます。

出力の構造

- vinfo (バージョン情報): バージョン構造体
- network_name (ネットワーク名): 文字列
- report_time (レポート作成時刻): ISO 8601 形式の UTC 時刻
- prefixes (プレフィックス): 以下の情報の配列
  - routers (ルーター): ルーター構造体の配列
  - prefix_type (プレフィックスのタイプ): 文字列
  - prefix_area (プレフィックスの領域): 文字列
  - prefix (プレフィックス): プレフィックス構造体

例

```
#!/usr/bin/perl

if (!defined($ARGV[0]) || !defined($ARGV[1])) {
   printf "usage: RouteAnalyzer.api_prefix_list_filtered ip
      database\n";
   exit(0);
}

my $rexip = $ARGV[0];
my $database = $ARGV[1];

use strict;
use RPC::XML::Client;
use RPC::XML 'time2iso8601';
use Date::Parse;
use Data::Dumper; $Data::Dumper::Terse = 1; $Data::Dumper::
   Indent = 1;
my $client;
my $req;
my @reqs;
my $password = 'admin';
$client = new RPC::XML::Client "http://$rexip:2000/RPC2";

my $t1 = str2time("28 Feb 2005 15:50:22 PST");

push (@reqs,
   RPC::XML::request->new('RouteAnalyzer.api_prefix_list_filtered',
      RPC::XML::RPC_STRING($password),
      RPC::XML::RPC_STRING($database),
      RPC::XML::datetime_iso8601->new(time2iso8601($t1)),
      new RPC::XML::struct (ip4_addr => "192.168.120.120")
   )
);

foreach (@reqs) {
   my $res = $client->send_request($_);
   if ($res->is_fault) {print("---XMLRPC FAULT ---"); }
   my $value1 = $res->value;
   print (STDERR join "\n", "---XMLRPC RESULT value1---",
      Dumper ($value1) );
}
```

サンプル出力

```
---XMLRPC RESULT value1 ---
{
   'vinfo' => {
      'software_version' => 'Build 4.0.90 HP RAMS',
      'appliance_version' => '0.5.24'
   },
   'network_name' => 'Lab',
   'report_time' => '20051031T22:58:04',
   'prefixes' => [
      {
         'routers' => [
            {
               'nodeType' => 'ASBR',
               'ip_addr' => {'ip4_addr' => '192,168,120,120'},
               'nodeState' => 'DOWN',
               'nodeProto' => 'Static',
               'name' => 'Router16',
               'nodeArea' => 'Lab.EIGRP/AS1',
               'maskLen' => '32',
               'systemID' => '192,168,120,120'
            },
            {
               'nodeType' => 'Internal',
               'ip_addr' => {'ip4_addr' => '192.168.220.20'},
               'nodeState' => 'DOWN',
               'nodeProto' => 'Static',
               'name' => 'Router20',
               'nodeArea' => 'Lab.EIGRP/AS1',
               'maskLen' => '32',
               'systemID' => '192.168.220.20'
            },
         ],
         'prefix_type' => 'Static',
         'prefix_area' => 'AllStaticRoutes.Static',
         'prefix' => {
            'masklen' => '0',
            'ip_addr' => {
               'ip4_addr' => '0.0.0.0'
            }
         }
      }...
   ]
}
```

api_prefix_list_multi_orig

RPC 呼び出し：RouteAnalyzer.api_prefix_list_multi_orig {password} {database name} {time}

このクエリーは、指定されたネットワークの複数のルーターによって通知されたプレフィックスのリストを返します。このクエリーが返す結果は、api_prefix_list クエリーが返す結果のサブセットです。

入力パラメータ

- **password**: クエリーに設定されたパスワード。
- **database name**: データベース階層に含まれる 1 つ以上のデータベース名 (複数の場合はスペースで区切る)。名前は、下にサブツリーを含む管理ドメイン名 (CorpNet など) か、データベース名全体 (CorpNet.EIGRP/AS100 など) です。
- **time**: UTC タイムゾーンの時刻を ISO 8601 形式で指定します (`20050725T21:47:35` など)。クエリーの結果は、指定された時刻のネットワークの状態に基づいて計算されます。

出力の構造

- vinfo (バージョン情報): バージョン構造体
- network_name (ネットワーク名): 文字列
- report_time (レポート作成時刻): ISO 8601 形式の UTC 時刻
- prefixes (プレフィックス): 以下の情報の配列
  - routers (ルーター): ルーター構造体の配列
  - prefix_type (プレフィックスのタイプ): 文字列
  - prefix_area (プレフィックスの領域): 文字列
  - prefix (プレフィックス): プレフィックス構造体

例

```
#!/usr/bin/perl

if (!defined($ARGV[0]) || !defined($ARGV[1])) {
   printf "usage: RouteAnalyzer.api_prefix_list_multi_orig ip
      database\n";
   exit(0);
}

my $rexip = $ARGV[0];
my $database = $ARGV[1];

use strict;
use RPC::XML::Client;
use RPC::XML 'time2iso8601';
use Date::Parse;
use Data::Dumper; $Data::Dumper::Terse = 1; $Data::Dumper::
   Indent = 1;
my $client;
my $req;
my @reqs;
my $password = 'admin';
$client = new RPC::XML::Client "http://$rexip:2000/RPC2";

my $t1 = 1058927123;

push (@reqs,
   RPC::XML::request->new('RouteAnalyzer.api_prefix_list_multi_
      orig',
      RPC::XML::RPC_STRING($password),
      RPC::XML::RPC_STRING($database),
      RPC::XML::datetime_iso8601->new(time2iso8601($t1))
   )
);

foreach (@reqs) {
   my $res = $client->send_request($_);
   if ($res->is_fault) {print("---XMLRPC FAULT ---"); }
   my $value1 = $res->value;
   print (STDERR join "\n", Dumper($value1) );
}
```

サンプル出力

```
{
   'vinfo' => {
      'software_version' => 'Build 4.0.90 HP RAMS',
      'appliance_version' => '0.5.24'
   },
   'network_name' => 'pd353',
   'report_time' => '20051028T00:45:15',
   'prefixes' => [
      {
         'routers' => [
            {
               'nodeType' => 'ASBR',
               'ip_addr' => {'ip4_addr' => '192,168,120,120'},
               'nodeState' => 'DOWN',
               'nodeProto' => 'Static',
               'name' => 'Router16',
               'nodeArea' => 'pd353.Left.EIGRP/AS1',
               'maskLen' => '32',
               'systemID' => '192,168,120,120'
            },
            {
               'nodeType' => 'ASBR',
               'ip_addr' => {'ip4_addr' => '192,168,122,122'},
               'nodeState' => 'DOWN',
               'nodeProto' => 'Static',
               'name' => 'Router26',
               'nodeArea' => 'pd353.Left.EIGRP/AS1',
               'maskLen' => '32',
               'systemID' => '192,168,122,122'
            },
            {
               'nodeType' => 'Internal',
               'ip_addr' => {'ip4_addr' => '192.168.220.20'},
               'nodeState' => 'DOWN',
               'nodeProto' => 'Static',
               'name' => 'Router20',
               'nodeArea' => 'pd353.Left.EIGRP/AS1',
               'maskLen' => '32',
               'systemID' => '192.168.220.20'
            },
         ],
         'prefix_type' => 'Static',
         'prefix_area' => 'AllStaticRoutes.Static',
```

```
            'prefix' => {
               'masklen' => '0',
               'ip_addr' => {
                  'ip4_addr' => '0.0.0.0'
               }
            }
         }...
      ]
   }
```

api_prefix_list_same

RPC 呼び出し： RouteAnalyzer.api_prefix_list_same {password} {database name} {time}

このクエリーは、指定されたデータベースによりネットワーク内で通知されたすべてのプレフィックスに関して、同一プレフィックスまたは個別のプレフィックスを通知するすべてのルーターを返します。内部のプレフィックス (IGP 固有) と外部のプレフィックス (別のルーティングプロトコルからインポートされる) は別々に比較されます。

入力パラメータ

- **password**: クエリーに設定されたパスワード。
- **database name**: データベース階層に含まれる 1 つ以上のデータベース名 (複数の場合はスペースで区切る)。名前は、下にサブツリーを含む管理ドメイン名 (CorpNet など) か、データベース名全体 (CorpNet.EIGRP/AS100 など) です。
- **time**: UTC タイムゾーンの時刻を ISO 8601 形式で指定します (`20050725T21:47:35` など)。クエリーの結果は、指定された時刻のネットワークの状態に基づいて計算されます。

出力の構造

- vinfo (バージョン情報): バージョン構造体
- network_name (ネットワーク名): 文字列
- report_time (レポート作成時刻): ISO 8601 形式の UTC 時刻
- prefixes (プレフィックス): 以下の情報の配列
  - prefix_type (プレフィックスのタイプ): 文字列
  - prefix_area (プレフィックスの領域): 文字列
  - prefix (プレフィックス): プレフィックス構造体
  - routers (ルーター): ルーター構造体の配列

例

```
#!/usr/bin/perl

use strict;
use RPC::XML::Client;
use RPC::XML 'time2iso8601';
use Date::Parse;
use Data::Dumper; $Data::Dumper::Terse = 1; $Data::Dumper::
   Indent = 1;

my $t1 = time2iso8601(str2time("17 Dec 2002 00:00:00 PST"));
my $request = RPC::XML::request->new(
   'RouteAnalyzer.api_prefix_list_same',
   RPC::XML::RPC_STRING( 'admin' ),   //password
   RPC::XML::RPC_STRING( 'CorpNet' ), //database name
   RPC::XML::datetime_iso8601->new($t1)
);
my $client = new RPC::XML::Client 'http://hostname:2000/RPC2';
my $result = $client->send_request($request);
if ($result->is_fault) { print("--- XMLRPC FAULT ---"); }
print(STDERR join "\n", "--- XMLRPC RESULT---",
   Dumper($result->value), '');
```

サンプル出力

```
--- XMLRPC RESULT ---
{
   'vinfo' => {
      'software_version' => 'Build 4.0.90 HP RAMS',
      'appliance_version' => '0.5.24',
   },
   'report_time' => '20030303T20:52:20',
   'network_name' => 'CorpNet',
   'prefixes' => [
      {
         'prefix_type' => 'Area-Ext',
         'prefix_area' => '00000001/ospf',
         'prefix' => {
            'ip_addr' => {'ip4_addr' => '192,168,200,200'},
            'masklen' => '24'
         },
         'routers' => [
            {
               'nodeProto' => 'ospf',
               'ip_addr' => {'ip4_addr' => '192,168,201,201'},
               'nodeType' => 'ASBR',
               'name' => '',
               'systemID' => '091001011001:00'
            },
            {'nodeProto'...'ip_addr'...'nodeType'...'name'
               ...'systemID' ...}
            {'nodeProto'...'ip_addr'...'nodeType'...'name'
               ...'systemID' ...}
         ] // end routers
      }, // end first prefix
      {'prefix_type'...'prefix_area'...'prefix'....'routers'
         => [...]},
      {'prefix_type'...'prefix_area'...'prefix'....'routers'
         => [...]}
   ] // end prefixes
}
```

api_resource_status

RPC 呼び出し：RouteAnalyzer.api_resource_status {password}

このクエリーは、アプライアンスのメモリ、ディスク、スワップ領域の現状を返します。使用領域、空き領域、全領域の容量以外に、ユーザーおよびシステムによる CPU 使用率、アイドル状態の CPU の比率、その他の用途で使用されている CPU の比率を表示します。

入力パラメータ

- **password**: クエリーに設定されたパスワード。

出力の構造

- vinfo (バージョン情報): バージョン構造体
- resouces (リソース):
  - memory (メモリ):
    - free (空き領域): 整数
    - used (使用領域): 整数
    - total (領域全体): 整数
    - pct (パーセント): double (現在のメモリ使用率)
  - disk (ディスク):
    - free (空き領域): 整数
    - used (使用領域): 整数
    - total (領域全体): 整数
    - pct (パーセント): double (現在のディスク使用率)
  - swap (スワップ):
    - free (空き領域): 整数
    - used (使用領域): 整数
    - total (領域全体): 整数

- pct (パーセント): double (現在のスワップ使用率)

- cpu (CPU):
  - user (ユーザー): double (パーセント)
  - system (システム): double (パーセント)
  - idle (アイドル): double (パーセント)
  - other (その他): double (パーセント。「other」は、nice 値が設定されたプロセス、I/O 待ち、ハードウェアおよびソフトウェア割り込みの処理など、他の目的で使用されている CPU の比率です)

例

```
#!/usr/bin/perl

if(!defined($ARGV[0])) {
   printf "usage:RouteAnalyzer.api_resource_status ip\n";
   exit(0);
}

my $rexip = $ARGV[0];

use strict;
use RPC::XML::Client;
use RPC::XML 'time2iso8601';
use Date::Parse;
use Data::Dumper; $Data::Dumper::Terse = 1; $Data::Dumper::
   Indent = 1;
my $client;
my $req;
my @reqs;
my $password = 'admin';

$client = new RPC::XML::Client "http://$rexip:2000/RPC2";

push (@reqs,
   RPC::XML::request->new('RouteAnalyzer.api_resource_status',
      RPC::XML::RPC_STRING($password)
   )
);

foreach (@reqs) {
   my $res = $client->send_request($_);
   if ($res->is_fault) {print("---XMLRPC FAULT ---"); }
   my $value1 = $res->value;
   print Dumper($value1);
}
```

サンプル出力

```
{
   'vinfo' => {
      'software_version' => 'Build 4.0.90 HP RAMS',
      'appliance_version' => '0.5.24'
   },
   'resources' => {
      'memory' => {
         'pct' => '71.70806685821979',
         'free' => '293032',
         'used' => '742712',
         'total' => '1035744'
      },
      'disk' => {
         'pct' => '79.65773029037497',
         'free' => '5195956',
         'used' => '27265044',
         'total' => '34227744'
      },
      'cpu' => {
         'system' => '0',
         'other' => '0',
         'user' => '0',
         'idle' => '100'
      },
      'swap' => {
         'pct' => '2.690779962482124',
         'free' => '2985840',
         'used' => '82564',
         'total' => '3068404'
      }
   }
}
```

api_router_events

RPC 呼び出し : RouteAnalyzer.api_router_events {password} {database name} {router}

このクエリーは、指定されたルーターに関連するイベントのリストを返します。

> **注記** このクエリーは、将来のリリースで廃止される予定のため推奨できません。新しいアプリケーションには、api_mp_events を使用してください。

入力パラメータ

- **password**: クエリーに設定されたパスワード。
- **database name**: データベース階層に含まれる 1 つ以上のデータベース名 (複数の場合はスペースで区切る)。名前は、下にサブツリーを含む管理ドメイン名 (CorpNet など) か、データベース名全体 (CorpNet.EIGRP/AS100 など) です。
- **router**: この XML 構造体には、ルーター ID またはルーターのインタフェースアドレスを示す IPv4 アドレスと、マスク長 32 を指定します。このマスクアドレスは、下位互換のために指定しますが、このクエリーでは無視されます。

出力の構造

- vinfo (バージョン情報): バージョン構造体
- network_name (ネットワーク名): 文字列
- report_time (レポート作成時刻): ISO 8601 形式の UTC 時刻
- events (イベント): 以下の情報の配列
  - router event (ルーターイベント):
    - eventType (イベントタイプ): 文字列
    - time (時刻): ISO 8601 形式の UTC 時刻
    - usec (マイクロ秒): 整数
    - router (ルーター): ルーター構造体

例

```
#!/usr/bin/perl

if (!defined($ARGV[0]) || !defined($ARGV[1])) {
   printf "usage:RouteAnalyzer.api_router_events ip database\n";
   exit(0);
}

my $rexip = $ARGV[0];
my $database = $ARGV[1];

use strict;
use RPC::XML::Client;
use RPC::XML 'time2iso8601';
use Date::Parse;
use Data::Dumper; $Data::Dumper::Terse = 1; $Data::Dumper::
   Indent = 1;
my $client;
my $req;
my @reqs;
my $password = 'admin';
$client = new RPC::XML::Client "http://$rexip:2000/RPC2";

push (@reqs,
   RPC::XML::request->new('RouteAnalyzer.api_router_events',
      RPC::XML::RPC_STRING($password),
      RPC::XML::RPC_STRING($database),
      new RPC::XML::struct(ip_addr =>
         new RPC::XML::struct(ip4_addr => "128.32.1.211"))
   )
);

foreach (@reqs) {
   my $res = $client->send_request($_);
   if ($res->is_fault) {print("---XMLRPC FAULT ---"); }
   my $value1 = $res->value;
   print (STDERR join "\n", "---XMLRPC RESULT value1---",
      Dumper ($value1) );
}
```

サンプル出力

```
---XMLRPC RESULT value1 ---
{
   'vinfo' => {
      'software_version' => 'Build 4.0.90 HP RAMS',
      'appliance_version' => '0.5.24'
   },
   'network_name' => 'UCBJul03a',
   'report_time' => '20051027T19:01:13',
   'events' => [
      {
         'router_event' => {
            'eventType' => 'advertised',
            'time' => '20030722T22:10:56:00',
            'usec' => '581591',
            'router' => {
               'nodeType' => 'ASBR',
               'ip_addr' => {'ip4_addr' => '128.32.1.211'},
               'nodeState' => 'DOWN',
               'nodeProto' => 'OSPF',
               'name' => '',
               'nodeArea' => 'UCBJul03a.OSPF/169,229,128,128',
               'maskLen' => '32',
               'systemID' => '128.32.1.211'
            }
         }
      },
      {
         'router_event' => {
            'eventType' => 'advertised',
            'time' => '20030722T22:10:52:00',
            'usec' => '851696',
            'router' => {
               'nodeType' => 'ASBR',
               'ip_addr' => {
                  'ip4_addr' => '128.32.1.211'
               },
               'nodeState' => 'DOWN',
               'nodeProto' => 'OSPF',
               'name' => '',
               'nodeArea' => 'UCBJul03a.OSPF/169,229,128,168',
               'maskLen' => '32',
               'systemID' => '128.32.1.211'
            }
```

```
            }
        },...
    ]
}
```

api_router_list

RPC 呼び出し：RouteAnalyzer.api_router_list {password} {database name} {time}

このクエリーは、指定されたネットワークのルーターリストを返します。

> **注記** このクエリーは、将来のリリースで廃止される予定のため推奨できません。新しいアプリケーションには、api_mp_routers を使用してください。

入力パラメータ

- **password**: クエリーに設定されたパスワード。
- **database name**: データベース階層に含まれる 1 つ以上のデータベース名 (複数の場合はスペースで区切る)。名前は、下にサブツリーを含む管理ドメイン名 (CorpNet など) か、データベース名全体 (CorpNet.EIGRP/AS100 など) です。
- **time**: UTC タイムゾーンの時刻を ISO 8601 形式で指定します (`20050725T21:47:35` など)。クエリーの結果は、指定された時刻のネットワークの状態に基づいて計算されます。

出力の構造

- vinfo (バージョン情報): バージョン構造体
- network_name (ネットワーク名): 文字列
- report_time (レポート作成時刻): ISO 8601 形式の UTC 時刻
- routers (ルーター): ルーター構造体の配列

例

```
#!/usr/bin/perl

if (!defined($ARGV[0]) || !defined($ARGV[1])) {
   printf "usage:RouteAnalyzer.api_router_list ip database\n";
   exit(0);
}

my $rexip = $ARGV[0];
my $database = $ARGV[1];

use strict;
use RPC::XML::Client;
use RPC::XML 'time2iso8601';
use Date::Parse;
use Data::Dumper; $Data::Dumper::Terse = 1; $Data::Dumper::
   Indent = 1;
my $client;
my $req; my @reqs;
my $password = 'admin';
$client = new RPC::XML::Client "http://$rexip:2000/RPC2";

my $t1 = str2time("24 Jul 2003 11:00:00 PST");

push (@reqs,
   RPC::XML::request->new('RouteAnalyzer.api_router_list',
      RPC::XML::RPC_STRING($password),
      RPC::XML::RPC_STRING($database),
      RPC::XML::datetime_iso8601->new(time2iso8601($t1))
   )
);

foreach (@reqs) {
   my $res = $client->send_request($_);
   if ($res->is_fault) {print("---XMLRPC FAULT ---"); }
   my $value1 = $res->value;
   print (STDERR join "\n", "---XMLRPC RESULT value1---",
      Dumper ($value1) );
}
```

サンプル出力

```
---XMLRPC RESULT value1 ---
{
   'vinfo' => {
      'software_version' => 'Build 4.0.90 HP RAMS',
      'appliance_version' => '0.5.24'
   },
   'routers' => [
      {
         'nodeType' => 'Internal',
         'ip_addr' => {'ip4_addr' => '169.229.128.130'},
         'nodeState' => 'UP',
         'nodeProto' => 'OSPF',
         'name' => '',
         'nodeArea' => 'UCBJul03a.OSPF/169,229,128,128',
         'maskLen' => '32',
         'systemID' => '169,229,128,130'
      },
      {
         'nodeType' => 'AreaBR_ASBR',
         'ip_addr' => {'ip4_addr' => '128.32.1.209'},
         'nodeState' => 'UP',
         'nodeProto' => 'OSPF',
         'name' => '',
         'nodeArea' => 'UCBJul03a.OSPF/169,229,128,128',
         'maskLen' => '32',
         'systemID' => '128.32.1.209'
      },
      {
         'nodeType' => 'Internal',
         'ip_addr' => {'ip4_addr' => '169.229.2.66'},
         'nodeState' => 'UP',
         'nodeProto' => 'OSPF',
         'name' => '',
         'nodeArea' => 'UCBJul03a.OSPF/169,229,128,168',
         'maskLen' => '32',
         'systemID' => '169.229.2.66'
      },
   ],
   'network_name' => 'UCBJul03a',
   'report_time' => '20051028T00:00:05'
}
```

api_router_summarizable

RPC 呼び出し：RouteAnalyzer.api_router_summarizable {password} {database name} {time}

このクエリーは、単一のプレフィックスとして集約可能な複数のプレフィックスを指定の時刻に通知しているルーターのリストを返します。RAMS は、これらのルーターごとに、集約した場合の集約後のプレフィックスと、その構成要素となるプレフィックスのリストを返します。内部の (IGP 固有の) プレフィックスと外部の (別のルーティングプロトコルからインポートされる) プレフィックスは、別のプレフィックスとみなされます。

入力パラメータ

- **password**: クエリーに設定されたパスワード。
- **database name**: データベース階層に含まれる 1 つ以上のデータベース名 (複数の場合はスペースで区切る)。名前は、下にサブツリーを含む管理ドメイン名 (CorpNet など) か、データベース名全体 (CorpNet.EIGRP/AS100 など) です。
- **time**: UTC タイムゾーンの時刻を ISO 8601 形式で指定します (`20050725T21:47:35` など)。クエリーの結果は、指定された時刻のネットワークの状態に基づいて計算されます。

出力の構造

- vinfo (バージョン情報): バージョン構造体
- report_time (レポート作成時刻): ISO 8601 形式の UTC 時刻
- network_name (ネットワーク名): 文字列
- routers (ルーター): 以下の情報の配列
  - router (ルーター): ルーター構造体
  - summarizable_prefixes (集約可能なプレフィックス): 以下の情報の配列
    - summary (集約後のプレフィックス): プレフィックス IP 構造体
    - contributors (構成要素): プレフィックス IP 構造体の配列

例

```
#!/usr/bin/perl

use strict;
use RPC::XML::Client;
use RPC::XML 'time2iso8601';
use Date::Parse;
use Data::Dumper; $Data::Dumper::Terse = 1; $Data::Dumper::
   Indent = 1;

my $t1 = time2iso8601(time);
my $request = RPC::XML::request->new(
   'RouteAnalyzer.api_router_summarizable',
   RPC::XML::RPC_STRING( 'admin' ),   //password
   RPC::XML::RPC_STRING( 'CorpNet' ), //database name
   RPC::XML::datetime_iso8601->new($t1)
);
my $client = new RPC::XML::Client 'http://hostname:2000/RPC2';
my $result = $client->send_request($request);
if ($result->is_fault) { print("--- XMLRPC FAULT ---"); }
print(STDERR join "\n", "--- XMLRPC RESULT---",
   Dumper($result->value), '');
```

サンプル出力

```
--- XMLRPC RESULT ---
{
   'vinfo' => {
      'software_version' => 'Build 4.0.90 HP RAMS'
      'appliance_version' => '0.5.24',
   },
   'report_time' => '20030303T21:09:29',
   'network_name' => 'CorpNet',
   'routers' => [
      {
         'router' => {
            'nodeProto' => 'ospf',
            'ip_addr' => {
               'ip4_addr' => '192,168,140,140'
            },
            'nodeType' => 'AreaBR',
            'name' => '',
            'systemID' => '004001001012:00'
         },
         'summarizable_prefixes' => [
               {'summary' => {
                  'ip_addr' => {'ip4_addr' => '192,168,150,150'},
                  'masklen' => '31'
               }, // end summary
               'contributors' => [
                  {
                     'ip_addr' => {'ip4_addr' => '192,168,150,150'},
                     'masklen' => '32'
                  },
                  {
                     'ip_addr' => {'ip4_addr' => '192,168,150,151'},
                     'masklen' => '32'
                  }
               ] // end contributors
            },
            {'summary' ...'contributors' },
            {'summary' ...'contributors' }
         ] // end summarizable_prefixes
      }, // end first router
      {'router' => {...}, 'summarizable_prefixes' => [...]},
      {'router' => {...}, 'summarizable_prefixes' => [...]}
   ] // end routers
}
```

api_system_health

RPC 呼び出し：RouteAnalyzer.api_system_health {password}

このクエリーは、ネットワーク内の Route Explorer システムと Traffic Explorer システムの稼働状態を返します。設定された各記録プロセスの記録ステータスとプロセスのデータベースの書き込みステータスも含まれます。マスター以外の装置はローカル装置の状態しか取得できませんが、マスター装置は各クライアントの状態を取得できます。

注記 クライアントには、マスターと同じクエリーパスワードが必要です。

入力パラメータ

- **password**: クエリーに設定されたパスワード。

出力の構造

- vinfo (バージョン情報): バージョン構造体
- units (装置): 以下の情報の配列
  - reachable (到達可能): ブール
  - ipaddr (IP アドレス): 文字列
  - processes (プロセス): 以下の情報の配列
    - globaldbname (グローバルデータベース名): 文字列
    - running (実行中): ブール
    - process (プロセス): 文字列
    - dbs (データベース): 以下の情報の配列
    - dbname (データベース名): 文字列
    - messages (メッセージ): 以下の情報の配列
    - msg (メッセージ): 文字列
    - last_write_time (最新書き込み時刻): ISO 8601 形式の UTC 時刻

例

```
if(!defined($ARGV[0])) {
   printf "usage:RouteAnalyzer.api_system_health ip\n";
   exit(0);
}

my $rexip = $ARGV[0];

use strict;
use RPC::XML::Client;
use RPC::XML 'time2iso8601';
use Date::Parse;
use Data::Dumper; $Data::Dumper::Terse = 1; $Data::Dumper::
   Indent = 1;
my $client;
my $req;
my @reqs;
my $password = 'admin';
$client = new RPC::XML::Client "http://$rexip:2000/RPC2";

push (@reqs,
   RPC::XML::request->new('RouteAnalyzer.api_system_health',
      RPC::XML::RPC_STRING($password)
   )
);

foreach (@reqs) {
   my $res = $client->send_request($_);
   if ($res->is_fault) {print("---XMLRPC FAULT ---"); }
   my $value1 = $res->value;
   print Dumper($value1);
}
```

サンプル出力

```
{
   'vinfo' => {
      'software_version' => 'Build 4.0.90 HP RAMS',
      'appliance_version' => '0.5.24'
   },
   'units' => [
      {
         'reachable' => '0',
         'processes' => [],
         'ipaddr' => '192.168.3.44'
      },
      {
         'reachable' => '1',
         'processes' => [
            {
               'label' => 'Traffic1',
               'running' => '0',
               'dbs' => [],
               'process' => 'Flow Recorder'
            }
         ],
         'ipaddr' => '192.168.3.126'
      },
      {
         'reachable' => '1',
         'processes' => [
            {
               'globaldbname' => 'JustBGP',
               'running' => '1',
               'dbs' => [
                  {
                     'messages' => [
                        {
                           'msg' => 'BGP Recorder is running'
                        },
                        {
                           'msg' => '1 of 1 peers established'
                        }
                     ],
                     'dbname' => 'JustBGP.BGP/AS65522',
                     'last_write_time' => '20061208T21:42:21'
                  }
               ],
```

```
                'process' => 'BGP Recorder'
              }
            ],
            'ipaddr' => '192.168.3.144'
          }
      ]
}
```

api_vpn_cust_rt_list

RPC 呼び出し：RouteAnalyzer.api_vpn_cust_rt_list {password} {database name} {operation} {customer name} {route target}

このクエリーは、指定されたデータベースに存在する VPN カスタマ名と RT (Route Target) のマッピングリストを返します。`get` 操作を指定してこのクエリーを実行すると、マッピングリストに変更は行われません。このクエリーでは、以下に示すように、マッピングリストを返す以外に、リストに変更を加える操作 (`add`、`del`、`reset`) も実行できます。

入力パラメータ

- **password**: クエリーに設定されたパスワード。
- **database name**: 管理ドメイン名 (CorpNet など) を指定してその下のサブツリーに含まれる VPN データベースを選択するか、データベース名全体 (CorpNet.BGP/AS65522/VPN など) を指定します。
- **operation**: 実行する具体的な操作。値は文字列であり、マッピングリストを返す場合は値「`get`」を、VPN カスタマを追加する場合は「`add`」を、VPN カスタマを削除する場合は「`del`」を、マッピングをすべて削除する場合は「`reset`」を使用します。
- **customer name**: `get` および `reset` 操作の場合は空文字列を指定し、`add` および `del` 操作の場合は VPN カスタマの名前を指定します。
- **route target**: `get` および `reset` 操作の場合は空文字列を指定し、`add` および `del` 操作の場合は RT の名前を指定します。

出力の構造

- vinfo (バージョン情報): バージョン構造体
- network_name (ネットワーク名): 文字列
- vpn_cust_rts (VPN カスタマと RT): 以下の情報の配列
  - name (名前): 文字列
  - rt (RT): 文字列

例

```
#!/usr/bin/perl

if (!defined($ARGV[0]) || !defined($ARGV[1])) {
   printf "usage: RouteAnalyzer.api_vpn_cust_rt_list ip
      database\n";
   exit(0);
}

my $rexip = $ARGV[0];
my $database = $ARGV[1];

use strict;
use RPC::XML::Client;
use RPC::XML 'time2iso8601';
use Date::Parse;
use Data::Dumper; $Data::Dumper::Terse = 1; $Data::Dumper::
   Indent = 1;
my $client;
my $req;
my @reqs;
my $password = 'admin';
$client = new RPC::XML::Client "http://$rexip:2000/RPC2";

push (@reqs,
   RPC::XML::request->new('RouteAnalyzer.api_vpn_cust_rt_list',
      RPC::XML::RPC_STRING($password),
      RPC::XML::RPC_STRING($database),
      RPC::XML::RPC_STRING('get'),
      RPC::XML::RPC_STRING(''),
      RPC::XML::RPC_STRING('')
   )
);

foreach (@reqs) {
   my $res = $client->send_request($_);
   if ($res->is_fault) {print("---XMLRPC FAULT ---"); }
   my $value1 = $res->value;
}
```

サンプル出力

```
{
   'vinfo' => {
      'software_version' => 'Build 4.0.90 HP RAMS',
      'appliance_version' => '0.5.24'
   },
   'network_name' => 'CorpNet',
   'vpn_cust_rts' => [
      {
         'name' => 'Customer1',
         'rt' => 'RT:65535:101'
      },
      {
         'name' => 'Customer2',
         'rt' => 'RT:65533:101'
      }
   ]
}
```

api_vpn_customer_pe_participation

RPC 呼び出し：RouteAnalyzer.api_vpn_customer_pe_participation {password} {database name} {time} {filter}

このクエリーは、各 VPN カスタマに関連する PE の統計を返します。

入力パラメータ

- **password**: クエリーに設定されたパスワード。
- **database name**: データベース階層に含まれる 1 つ以上のデータベース名 (複数の場合はスペースで区切る)。名前は、下にサブツリーを含む管理ドメイン名 (CorpNet など) か、データベース名全体 (CorpNet.BGP/AS65522/VPN など) です。
- **time**: UTC タイムゾーンの時刻を ISO 8601 形式で指定します (`20050725T21:47:35` など)。クエリーの結果は、指定された時刻のネットワークの状態に基づいて計算されます。
- **filter**: フィルターパラメータに一致するサブセットのみを出力するためのフィルター式。フィルター式の詳細は、『ユーザーガイド』の「History Navigator」の章の「式の構文」を参照してください。フィルター「any」を使用すると、結果がすべて返されます。

出力の構造

- vinfo (バージョン情報): バージョン構造体
- numReturnedEntries (返されたエントリの数): 50
- network_name (ネットワーク名): 文字列
- report_time (レポート作成時刻): ISO 8601 形式の UTC 時刻
- totalEntries (エントリの総数): 整数
- result (結果): 以下の情報の配列
  - customer (カスタマ): 文字列
  - numActivePEs (アクティブな PE の数): 整数
  - deviation (偏差): 整数

- numNewPEs (新しい PE の数): 整数
- numDownPEs (ダウンしている PE の数): 整数
- definition (定義): 文字列
- numBaselinePEs (基準時の PE の数): 整数

例

```
#!/usr/bin/perl

if (!defined($ARGV[0]) || !defined($ARGV[1])) {
   printf "usage: RouteAnalyzer.api_vpn_customer_pe_participation
      ip database\n";
   exit(0);
}

my $rexip = $ARGV[0];
my $database = $ARGV[1];
my $filter = "any";
$filter = $ARGV[2] if ($#ARGV >= 2);

use strict;
use RPC::XML::Client;
use RPC::XML 'time2iso8601';
use Date::Parse;
use Data::Dumper; $Data::Dumper::Terse = 1; $Data::Dumper::
   Indent = 1;
my $client;
my $req;
my @reqs;
my $password = 'admin';
$client = new RPC::XML::Client "http://$rexip:2000/RPC2";

my $t1 = str2time("28 Feb 2005 15:50:22 PST");

push (@reqs,
   RPC::XML::request->new('RouteAnalyzer.api_vpn_customer_pe_
      participation',
      RPC::XML::RPC_STRING($password),
      RPC::XML::RPC_STRING($database),
      RPC::XML::datetime_iso8601->new(time2iso8601($t1)),
      RPC::XML::RPC_STRING($filter)
   )
);

foreach (@reqs) {
   my $res = $client->send_request($_);
   if ($res->is_fault) {print("---XMLRPC FAULT ---"); }
   my $value1 = $res->value;
   print Dumper($value1);
}
```

サンプル出力

```
---XMLRPC RESULT value1 ---
{
   'vinfo' => {
      'software_version' => 'Build 4.0.90 HP RAMS',
      'appliance_version' => '0.5.24'
   },
   'numReturnedEntries' => '50',
   'network_name' => 'VOD',
   'report_time' => '20051115T19:19:00:00',
   'totalEntries' => '50',
   'result' => [
      {
         'customer' => 'Cust747',
         'numActivePEs' => '0',
         'deviation' => '100',
         'numNewPEs' => '0',
         'numDownPEs' => '0',
         'definition' => 'RT:600:1',
         'numBaselinePEs' => '0'
      }
   ]
}
```

api_vpn_customer_pe_list

RPC 呼び出し：RouteAnalyzer.api_vpn_customer_privacy {password} {database name} {time} {customer name} {filter}

このクエリーは、指定された VPN カスタマに関連する PE のリストを返します。

入力パラメータ

- **password**: クエリーに設定されたパスワード。
- **database name**: データベース階層に含まれる 1 つ以上のデータベース名 (複数の場合はスペースで区切る)。名前は、下にサブツリーを含む管理ドメイン名 (CorpNet など) か、データベース名全体 (CorpNet.BGP/AS65522/VPN など) です。
- **time**: UTC タイムゾーンの時刻を ISO 8601 形式で指定します (`20050725T21:47:35` など)。クエリーの結果は、指定された時刻のネットワークの状態に基づいて計算されます。
- **customer name**: PE リストが必要な VPN カスタマの名前。
- **filter**: フィルターパラメータに一致するサブセットのみを出力するためのフィルター式。フィルター式の詳細は、『ユーザーガイド』の「History Navigator」の章の「式の構文」を参照してください。フィルター「any」を使用すると、結果がすべて返されます。

出力の構造

- vinfo (バージョン情報): バージョン構造体
- numReturnedEntries (返されたエントリの数): 整数
- network_name (ネットワーク名): 文字列
- report_time (レポート作成時刻): ISO 8601 形式の UTC 時刻
- totalEntries (エントリの総数): 整数
- result (結果): 以下の情報の配列
  - PE: ルーター構造体
  - vpnState (VPN の状態): 状態構造体 (基準値あり)

例

```
#!/usr/bin/perl
if(!defined($ARGV[0]) || !defined($ARGV[1]) || !defined($ARGV[2]))
{
   printf "usage: RouteAnalyzer.api_vpn_customer_pe_list ip
      database customer\n";
   exit(0);
}
my $rexip = $ARGV[0];
my $database = $ARGV[1];
my $customer = $ARGV[2];
my $filter = "any";
$filter = $ARGV[3] if ($#ARGV >= 3);

use strict;
use RPC::XML::Client;
use RPC::XML 'time2iso8601';
use Date::Parse;
use Data::Dumper; $Data::Dumper::Terse = 1; $Data::Dumper::
   Indent = 1;
my $client;
my $req;
my @reqs;
my $password = 'admin';
$client = new RPC::XML::Client "http://$rexip:2000/RPC2";
my $t1 = str2time("30 Aug 2005 00:26:30 PDT");

push (@reqs,
   RPC::XML::request->new('RouteAnalyzer.api_vpn_customer_pe_list',
      RPC::XML::RPC_STRING($password),
      RPC::XML::RPC_STRING($database),
      RPC::XML::datetime_iso8601->new(time2iso8601($t1)),
      RPC::XML::RPC_STRING($customer),
      RPC::XML::RPC_STRING($filter)
   )
);

foreach (@reqs) {
   my $res = $client->send_request($_);
   if ($res->is_fault) {print("---XMLRPC FAULT ---"); }
   my $value1 = $res->value;
   print (STDERR join "\n", "---XMLRPC RESULT value1---",
      Dumper ($value1) );
}
```

サンプル出力

```
---XMLRPC RESULT value1 ---
{
   'vinfo' => {
      'software_version' => 'Build 4.0.90 HP RAMS',
      'appliance_version' => '0.5.24'
   },
   'numReturnedEntries' => '1',
   'network_name' => 'VOD',
   'report_time' => '20051115T19:14:20:00',
   'totalEntries' => '1',
   'result' => [
      {
         'PE' => {
            'type' => 'Originator',
            'ipaddr' => '192,168,180,180'
         },
         'vpnState' => {
            'inBaseline' => 'false',
            'down' => 'true'
         }
      }
   ]
}
```

api_vpn_customer_reachability

RPC 呼び出し：RouteAnalyzer.api_vpn_customer_reachability {password} {database name} {time} {filter}

このクエリーは、各 VPN カスタマの到達可能性統計を返します。到達可能性は、基準時の到達可能性からの偏差を示すパーセント値で示されます。たとえば、一部の経路がダウンしており、使用可能な経路が基準時よりも少ない場合、この値は負になります。基準時に認識されていなかった新しい経路が追加されると、値は正になります。

入力パラメータ

- **password**: クエリーに設定されたパスワード。
- **database name**: データベース階層に含まれる 1 つ以上のデータベース名 (複数の場合はスペースで区切る)。名前は、下にサブツリーを含む管理ドメイン名 (CorpNet など) か、データベース名全体 (CorpNet.BGP/AS65522/VPN など) です。
- **time**: UTC タイムゾーンの時刻を ISO 8601 形式で指定します (`20050725T21:47:35` など)。クエリーの結果は、指定された時刻のネットワークの状態に基づいて計算されます。
- **filter**: フィルターパラメータに一致するサブセットのみを出力するためのフィルター式。フィルター式の詳細は、『ユーザーガイド』の「History Navigator」の章の「式の構文」を参照してください。フィルター「any」を使用すると、結果がすべて返されます。

出力の構造

- vinfo (バージョン情報): バージョン構造体
- numReturnedEntries (返されたエントリの数): 整数
- network_name (ネットワーク名): 文字列
- report_time (レポート作成時刻): ISO 8601 形式の UTC 時刻
- totalEntries (エントリの総数): 整数
- result (結果): 以下の情報の配列

- customer (カスタマ): 文字列
- definition (定義): 文字列
- numPEs (PE の数): 整数
- numActiveRoutes (アクティブな経路の数): 整数
- numBaselineRoutes (基準時の経路の数): 整数
- numDownRoutes (ダウンしている経路の数): 整数
- numNewRoutes (新しい経路の数): 整数
- deviation (偏差): 整数

例

```
#!/usr/bin/perl

if (!defined($ARGV[0]) || !defined($ARGV[1])) {
   printf "usage: RouteAnalyzer.api_vpn_customer_reachability ip
      database\n";
   exit(0);
}

my $rexip = $ARGV[0];
my $database = $ARGV[1];
my $filter = "any";

$filter = $ARGV[2] if ($#ARGV >= 2);

use strict;
use RPC::XML::Client;
use RPC::XML 'time2iso8601';
use Date::Parse;
use Data::Dumper; $Data::Dumper::Terse = 1; $Data::Dumper::
   Indent = 1;
my $client;
my $req;
my @reqs;
my $password = 'admin';
$client = new RPC::XML::Client "http://$rexip:2000/RPC2";
my $t1 = str2time("28 Feb 2005 15:50:22 PST");

push (@reqs,
   RPC::XML::request->new('RouteAnalyzer.api_vpn_customer_
      reachability',
      RPC::XML::RPC_STRING($password),
      RPC::XML::RPC_STRING($database),
      RPC::XML::datetime_iso8601->new(time2iso8601($t1)),
      RPC::XML::RPC_STRING($filter)
   )
);

foreach (@reqs) {
   my $res = $client->send_request($_);
   if ($res->is_fault) {print("---XMLRPC FAULT ---"); }
   my $value1 = $res->value;
   print Dumper($value1);
}
```

サンプル出力

```
---XMLRPC RESULT value1 ---
{
   'vinfo' => {
      'software_version' => 'Build 4.0.90 HP RAMS',
      'appliance_version' => '0.5.24'
   },
   'numReturnedEntries' => '50',
   'network_name' => 'VOD',
   'report_time' => '20051115T19:19:49:00',
   'totalEntries' => '50',
   'result' => [
      {
         'numDownRoutes' => '1',
         'numActiveRoutes' => '0',
         'numNewRoutes' => '0',
         'numPEs' => '0',
         'customer' => 'Cust747',
         'deviation' => '100',
         'numBaselineRoutes' => '1',
         'definition' => 'RT:600:1'
      }
   ]
}
```

api_vpn_customer_reachability_by_peer

RPC 呼び出し：RouteAnalyzer.api_vpn_customer_reachability_by_peer {password} {database name} {time} {customer name} {filter}

このクエリーは、指定された VPN カスタマの各 PE の到達可能性統計を返します。到達可能性は、基準時の到達可能性からの偏差を示すパーセント値で示されます。たとえば、一部の経路がダウンしており、使用可能な経路が基準時よりも少ない場合、この値は負になります。基準時に認識されていなかった新しい経路が追加されると、値は正になります。

入力パラメータ

- **password**: クエリーに設定されたパスワード。
- **database name**: データベース階層に含まれる 1 つ以上のデータベース名 (複数の場合はスペースで区切る)。名前は、下にサブツリーを含む管理ドメイン名 (CorpNet など) か、データベース名全体 (CorpNet.BGP/AS65522/VPN など) です。
- **time**: UTC タイムゾーンの時刻を ISO 8601 形式で指定します (`20050725T21:47:35` など)。クエリーの結果は、指定された時刻のネットワークの状態に基づいて計算されます。
- **customer name**: 到達可能性情報が必要な VPN カスタマの名前。
- **filter**: フィルターパラメータに一致するサブセットのみを出力するためのフィルター式。フィルター式の詳細は、『ユーザーガイド』の「History Navigator」の章の「式の構文」を参照してください。フィルター「any」を使用すると、結果がすべて返されます。

出力の構造

- vinfo (バージョン情報): バージョン構造体
- numReturnedEntries (返されたエントリの数): 整数
- network_name (ネットワーク名): 文字列
- report_time (レポート作成時刻): ISO 8601 形式の UTC 時刻
- totalEntries (エントリの総数): 整数

- result (結果): 以下の情報の配列
  - PE: ルーター構造体
  - vpnState (VPN の状態): 状態構造体 (基準値あり)
  - numActiveRoutes (アクティブな経路の数): 整数
  - numBaselineRoutes (基準時の経路の数): 整数
  - numDownRoutes (ダウンしている経路の数): 整数
  - numNewRoutes (新しい経路の数): 整数
  - deviation (偏差): 整数

例

```
#!/usr/bin/perl

if(!defined($ARGV[0]) || !defined($ARGV[1]) || !defined($ARGV[2]))
{
   printf "usage: RouteAnalyzer.api_vpn_customer_reachability_by_
      peer ip database customer\n";
   exit(0);
}

my $rexip = $ARGV[0];
my $database = $ARGV[1];
my $customer = $ARGV[2];
my $filter = "any";

$filter = $ARGV[3] if ($#ARGV >= 3);

use strict;
use RPC::XML::Client;
use RPC::XML 'time2iso8601';
use Date::Parse;
use Data::Dumper; $Data::Dumper::Terse = 1; $Data::Dumper::
   Indent = 1;
my $client;
my $req;
my @reqs;
my $password = 'admin';
$client = new RPC::XML::Client "http://$rexip:2000/RPC2";

my $t1 = str2time("30 Aug 2005 00:26:30 PDT");

push (@reqs,
   RPC::XML::request->new('RouteAnalyzer.api_vpn_customer_
      reachability_by_peer',
      RPC::XML::RPC_STRING($password),
      RPC::XML::RPC_STRING($database),
      RPC::XML::datetime_iso8601->new(time2iso8601($t1)),
      RPC::XML::RPC_STRING($customer),
      RPC::XML::RPC_STRING($filter)
   )
);

foreach (@reqs) {
   my $res = $client->send_request($_);
```

```
    if ($res->is_fault) {print("---XMLRPC FAULT ---"); }
    my $value1 = $res->value;
    print (STDERR join "\n", "---XMLRPC RESULT value1---",
       Dumper ($value1) );
}
```

サンプル出力

```
---XMLRPC RESULT value1 ---
{
   'vinfo' => {
      'software_version' => 'Build 4.0.90 HP RAMS',
      'appliance_version' => '0.5.24'
   },
   'numReturnedEntries' => '25',
   'network_name' => 'VOD',
   'report_time' => '20051115T19:12:35:00',
   'totalEntries' => '25',
   'result' => [
      {
         'numDownRoutes' => '0',
         'numActiveRoutes' => '1',
         'numNewRoutes' => '0',
         'PE' => {
            'type' => 'Originator',
            'ipaddr' => '192,168,180,180'
         },
         'deviation' => '0',
         'numBaselineRoutes' => '1',
         'vpnState' => {
            'inBaseline' => 'false',
            'down' => 'true'
         }
      }
   ]
}
```

api_vpn_route_target_pe_participation

RPC 呼び出し：RouteAnalyzer.api_vpn_route_target_pe_participation {password} {database name} {time} {filter}

このクエリーは、指定されたネットワークの各 RT (Route Target) に関連する PE の統計を返します。この統計には、RT に関する情報、基準時からの偏差、アクティブな PE の数、ダウンしている PE の数、基準時後に新しく追加された PE の数が含まれます。

入力パラメータ

- **password**: クエリーに設定されたパスワード。
- **database name**: データベース階層に含まれる 1 つ以上のデータベース名 (複数の場合はスペースで区切る)。名前は、下にサブツリーを含む管理ドメイン名 (CorpNet など) か、データベース名全体 (CorpNet.BGP/AS65522/VPN など) です。
- **time**: UTC タイムゾーンの時刻を ISO 8601 形式で指定します (`20050725T21:47:35` など)。クエリーの結果は、指定された時刻のネットワークの状態に基づいて計算されます。
- **filter**: フィルターパラメータに一致するサブセットのみを出力するためのフィルター式。フィルター式の詳細は、『ユーザーガイド』の「History Navigator」の章の「式の構文」を参照してください。フィルター「any」を使用すると、結果がすべて返されます。

出力の構造

- vinfo (バージョン情報): バージョン構造体
- numReturnedEntries (返されたエントリの数): 整数
- network_name (ネットワーク名): 文字列
- report_time (レポート作成時刻): ISO 8601 形式の UTC 時刻
- totalEntries (エントリの総数): 整数
- result (結果): 以下の情報の配列
  - routeTarget (RT): 文字列

- numActivePEs (アクティブな PE の数): 整数
- numBaselinePEs (基準時の PE の数): 整数
- numDownPEs (ダウンしている PE の数): 整数
- numNewPEs (新しい PE の数): 整数
- deviation (偏差): 整数

例

```
#!/usr/bin/perl

if (!defined($ARGV[0]) || !defined($ARGV[1])) {
   printf "usage: RouteAnalyzer.api_vpn_route_target_pe_
      participation ip database\n";
   exit(0);
}

my $rexip = $ARGV[0];
my $database = $ARGV[1];
my $filter = "any";
$filter = $ARGV[2] if ($#ARGV >= 2);

use strict;
use RPC::XML::Client;
use RPC::XML 'time2iso8601';
use Date::Parse;
use Data::Dumper; $Data::Dumper::Terse = 1; $Data::Dumper::
   Indent = 1;
my $client;
my $req;
my @reqs;
my $password = 'admin';
$client = new RPC::XML::Client "http://$rexip:2000/RPC2";

my $t1 = str2time("28 Feb 2005 15:50:22 PST");

push (@reqs,
   RPC::XML::request->new('RouteAnalyzer.api_vpn_route_target_pe_
      participation',
      RPC::XML::RPC_STRING($password),
      RPC::XML::RPC_STRING($database),
      RPC::XML::datetime_iso8601->new(time2iso8601($t1)),
      RPC::XML::RPC_STRING($filter)
   )
);

foreach (@reqs) {
   my $res = $client->send_request($_);
   if ($res->is_fault) {print("---XMLRPC FAULT ---"); }
   my $value1 = $res->value;
   print Dumper($value1);
}
```

サンプル出力

```
---XMLRPC RESULT value1 ---
{
   'vinfo' => {
      'software_version' => 'Build 4.0.90 HP RAMS',
      'appliance_version' => '0.5.24'
   },
   'numReturnedEntries' => '50',
   'network_name' => 'pd353',
   'report_time' => '20051028T00:23:42',
   'totalEntries' => '50',
   'result' => [
      {'routeTarget' => 'RT:65522:600',
         'numActivePEs' => '3',
         'deviation' => '100',
         'numNewPEs' => '3',
         'numDownPEs' => '0',
         'numBaselinePEs' => '0'
      },
      {'routeTarget' => 'RT:65522:2300',
         'numActivePEs' => '1',
         'deviation' => '100',
         'numNewPEs' => '1',
         'numDownPEs' => '0',
         'numBaselinePEs' => '0'
      },
      {'routeTarget' => 'RT:65522:500',
         'numActivePEs' => '2',
         'deviation' => '100',
         'numNewPEs' => '2',
         'numDownPEs' => '0',
         'numBaselinePEs' => '0'
      },
      {'routeTarget' => 'RT:65522:1500',
         'numActivePEs' => '2',
         'deviation' => '100',
         'numNewPEs' => '2',
         'numDownPEs' => '0',
         'numBaselinePEs' => '0'
      },
   ]
}
```

api_vpn_route_target_pe_list

RPC 呼び出し：RouteAnalyzer.api_vpn_route_target_privacy_by_peer {password} {database name} {time} {route target} {filter}

このクエリーは、指定された RT (Route Target) に関連する PE ルーターとその VPN 状態のリストを返します。

入力パラメータ

- **password**: クエリーに設定されたパスワード。
- **database name**: データベース階層に含まれる 1 つ以上のデータベース名 (複数の場合はスペースで区切る)。名前は、下にサブツリーを含む管理ドメイン名 (CorpNet など) か、データベース名全体 (CorpNet.BGP/AS65522/VPN など) です。
- **time**: UTC タイムゾーンの時刻を ISO 8601 形式で指定します (`20050725T21:47:35` など)。クエリーの結果は、指定された時刻のネットワークの状態に基づいて計算されます。
- **route target**: 対象の RT を指定するラベル (RT:600:1 など)。
- **filter**: フィルターパラメータに一致するサブセットのみを出力するためのフィルター式。フィルター式の詳細は、『ユーザーガイド』の「History Navigator」の章の「式の構文」を参照してください。フィルター「any」を使用すると、結果がすべて返されます。

出力の構造

- vinfo (バージョン情報): バージョン構造体
- numReturnedEntries (返されたエントリの数): 整数
- network_name (ネットワーク名): 文字列
- report_time (レポート作成時刻): ISO 8601 形式の UTC 時刻
- totalEntries (エントリの総数): 整数
- result (結果): 以下の情報の配列
  - PE: ルーター構造体

— vpnState (VPN の状態): 状態構造体 (基準値あり)

例

```
#!/usr/bin/perl

if(!defined($ARGV[0]) || !defined($ARGV[1]) || !defined($ARGV[2]))
{
   printf "usage: RouteAnalyzer.api_vpn_route_target_pe_list ip
      database route-target\n";
   exit(0);
}
my $rexip = $ARGV[0];
my $database = $ARGV[1];
my $route_target = $ARGV[2];
my $filter = "any";
$filter = $ARGV[3] if ($#ARGV >= 3);
use strict;
use RPC::XML::Client;
use RPC::XML 'time2iso8601';
use Date::Parse;
use Data::Dumper; $Data::Dumper::Terse = 1; $Data::Dumper::
   Indent = 1;
my $client;
my $req;
my @reqs;
my $password = 'admin';
$client = new RPC::XML::Client "http://$rexip:2000/RPC2";
my $t1 = str2time("28 Aug 2005 15:50:22 PDT");

push (@reqs,
   RPC::XML::request->new('RouteAnalyzer.api_vpn_route_target_
      pe_list',
      RPC::XML::RPC_STRING($password),
      RPC::XML::RPC_STRING($database),
      RPC::XML::datetime_iso8601->new(time2iso8601($t1)),
      RPC::XML::RPC_STRING($route_target),
      RPC::XML::RPC_STRING($filter)
   )
);

foreach (@reqs) {
   my $res = $client->send_request($_);
   if ($res->is_fault) {print("---XMLRPC FAULT ---"); }
   my $value1 = $res->value;
   print (STDERR join "\n", "---XMLRPC RESULT value1---",
      Dumper ($value1) );
}
```

サンプル出力

```
---XMLRPC RESULT value1 ---
{
   'vinfo' => {
      'software_version' => 'Build 4.0.90 HP RAMS',
      'appliance_version' => '0.5.24'
   },
   'numReturnedEntries' => '25',
   'network_name' => 'VOD',
   'report_time' => '20051108T19:51:50',
   'totalEntries' => '25',
   'result' => [
      {
         'PE' => {
            'type' => 'Originator',
            'ipaddr' => '192,168,180,180'
         },
         'vpnState' => {
            'inBaseline' => 'false',
            'down' => 'true'
         }
      }
   ]
}
```

api_vpn_route_target_reachability

RPC 呼び出し：RouteAnalyzer.api_vpn_route_target_reachability {password} {database name} {time} {filter}

このクエリーは、指定されたネットワークの各 RT (Route Target) の到達可能性統計を返します。この統計には、基準時からの偏差に関する情報、ダウンしている経路の数、アクティブな経路の数、基準時後に新しく追加された経路の数が含まれます。

入力パラメータ

- **password**: クエリーに設定されたパスワード。
- **database name**: データベース階層に含まれる 1 つ以上のデータベース名 (複数の場合はスペースで区切る)。名前は、下にサブツリーを含む管理ドメイン名 (CorpNet など) か、データベース名全体 (CorpNet.BGP/AS65522/VPN など) です。
- **time**: UTC タイムゾーンの時刻を ISO 8601 形式で指定します (`20050725T21:47:35` など)。クエリーの結果は、指定された時刻のネットワークの状態に基づいて計算されます。
- **filter**: フィルターパラメータに一致するサブセットのみを出力するためのフィルター式。フィルター式の詳細は、『ユーザーガイド』の「History Navigator」の章の「式の構文」を参照してください。フィルター「any」を使用すると、結果がすべて返されます。

出力の構造

- vinfo (バージョン情報): バージョン構造体
- numReturnedEntries (返されたエントリの数): 整数
- network_name (ネットワーク名): 文字列
- report_time (レポート作成時刻): ISO 8601 形式の UTC 時刻
- totalEntries (エントリの総数): 整数
- result (結果): 以下の情報の配列
  - routeTarget (RT): 文字列

- numPEs (PE の数): 整数
- numActiveRoutes (アクティブな経路の数): 整数
- numBaselineRoutes (基準時の経路の数): 整数
- numDownRoutes (ダウンしている経路の数): 整数
- numNewRoutes (新しい経路の数): 整数
- deviation (偏差): 整数

例

```
#!/usr/bin/perl

if (!defined($ARGV[0]) || !defined($ARGV[1])) {
   printf "usage: RouteAnalyzer.api_vpn_route_target_reachability
      ip database\n";
   exit(0);
}

my $rexip = $ARGV[0];
my $database = $ARGV[1];
my $filter = "any";
$filter = $ARGV[2] if ($#ARGV >= 2);

use strict;
use RPC::XML::Client;
use RPC::XML 'time2iso8601';
use Date::Parse;
use Data::Dumper; $Data::Dumper::Terse = 1; $Data::Dumper::
   Indent = 1;
my $client;
my $req;
my @reqs;
my $password = 'admin';
$client = new RPC::XML::Client "http://$rexip:2000/RPC2";

my $t1 = str2time("28 Jul 2004 08:25:51 PST");

push (@reqs,
   RPC::XML::request->new('RouteAnalyzer.api_vpn_route_target_
      reachability',
      RPC::XML::RPC_STRING($password),
      RPC::XML::RPC_STRING($database),
      RPC::XML::datetime_iso8601->new(time2iso8601($t1)),
      RPC::XML::RPC_STRING($filter)
   )
);

foreach (@reqs) {
   my $res = $client->send_request($_);
   if ($res->is_fault) {print("---XMLRPC FAULT ---"); }
   my $value1 = $res->value;
   print Dumper($value1);
}
```

サンプル出力

```
---XMLRPC RESULT value1 ---
{
   'vinfo' => {
      'software_version' => 'Build 4.0.90 HP RAMS',
      'appliance_version' => '0.5.24'
   },
   'numReturnedEntries' => '50',
   'network_name' => 'pd353',
   'report_time' => '20051027T23:25:19',
   'totalEntries' => '50',
   'result' => [
      {
         'routeTarget' => 'RT:65522:100',
         'numDownRoutes' => '0',
         'numActiveRoutes' => '0',
         'numPEs' => '0',
         'numNewRoutes' => '0',
         'deviation' => '100',
         'numBaselineRoutes' => '0'
      },
      {
         'routeTarget' => 'RT:65522:600',
         'numDownRoutes' => '0',
         'numActiveRoutes' => '0',
         'numPEs' => '0',
         'numNewRoutes' => '0',
         'deviation' => '100',
         'numBaselineRoutes' => '0'
      },
      {
         'routeTarget' => 'RT:65522:2400',
         'numDownRoutes' => '0',
         'numActiveRoutes' => '0',
         'numPEs' => '0',
         'numNewRoutes' => '0',
         'deviation' => '100',
         'numBaselineRoutes' => '0'
      },
      {
         'routeTarget' => 'RT:65522:700',
         'numDownRoutes' => '0',
         'numActiveRoutes' => '0',
         'numPEs' => '0',
```

```
            'numNewRoutes' => '0',
            'deviation' => '100',
            'numBaselineRoutes' => '0'
        }
    ]
}
```

api_vpn_route_target_reachability_by_peer

RPC 呼び出し：RouteAnalyzer.api_vpn_route_target_reachability_by_peer {password} {database name} {time} {route target} {filter}

このクエリーは、指定された RT (Route Target) の各 PE の到達可能性統計を返します。この統計には、基準時からの偏差に関する情報、ダウンしている経路の数、アクティブな経路の数、基準時後に新しく追加された経路の数が含まれます。

入力パラメータ

- **password**: クエリーに設定されたパスワード。
- **database name**: データベース階層に含まれる 1 つ以上のデータベース名 (複数の場合はスペースで区切る)。名前は、下にサブツリーを含む管理ドメイン名 (CorpNet など) か、データベース名全体 (CorpNet.BGP/AS65522/VPN など) です。
- **time**: UTC タイムゾーンの時刻を ISO 8601 形式で指定します (`20050725T21:47:35` など)。クエリーの結果は、指定された時刻のネットワークの状態に基づいて計算されます。
- **route target**: 対象の RT を指定するラベル (RT:600:1 など)。
- **filter**: フィルターパラメータに一致するサブセットのみを出力するためのフィルター式。フィルター式の詳細は、『ユーザーガイド』の「History Navigator」の章の「式の構文」を参照してください。フィルター「any」を使用すると、結果がすべて返されます。

出力の構造

- vinfo (バージョン情報): バージョン構造体
- numReturnedEntries (返されたエントリの数): 整数
- network_name (ネットワーク名): 文字列
- report_time (レポート作成時刻): ISO 8601 形式の UTC 時刻
- totalEntries (エントリの総数): 整数
- result (結果): 以下の情報の配列
  - PE: ルーター構造体

- vpnState (VPN の状態): 状態構造体 (基準値あり)
- numActiveRoutes (アクティブな経路の数): 整数
- numBaselineRoutes (基準時の経路の数): 整数
- numDownRoutes (ダウンしている経路の数): 整数
- numNewRoutes (新しい経路の数): 整数
- deviation (偏差): 整数

例

```
#!/usr/bin/perl

if(!defined($ARGV[0]) || !defined($ARGV[1]) || !defined($ARGV[2]))
{
   printf "usage: RouteAnalyzer.api_vpn_route_target_reachability_
      by_peer ip database route_target\n";
   exit(0);
}

my $rexip = $ARGV[0];
my $database = $ARGV[1];
my $filter = "any";
my $route_target = $ARGV[2];

$filter = $ARGV[3] if ($#ARGV >= 3);

use strict;
use RPC::XML::Client;
use RPC::XML 'time2iso8601';
use Date::Parse;
use Data::Dumper; $Data::Dumper::Terse = 1; $Data::Dumper::
     Indent = 1;
my $client;
my $req;
my @reqs;
my $password = 'admin';
$client = new RPC::XML::Client "http://$rexip:2000/RPC2";

my $t1 = str2time("28 Aug 2005 16:16:45 PDT");

push (@reqs,
   RPC::XML::request->new('RouteAnalyzer.api_vpn_route_target_
      reachability_by_peer',
      RPC::XML::RPC_STRING($password),
      RPC::XML::RPC_STRING($database),
      RPC::XML::datetime_iso8601->new(time2iso8601($t1)),
      RPC::XML::RPC_STRING($route_target),
      RPC::XML::RPC_STRING($filter)
   )
);

foreach (@reqs) {
   my $res = $client->send_request($_);
```

```
   if ($res->is_fault) {print("---XMLRPC FAULT ---"); }
   my $value1 = $res->value;
   print (STDERR join "\n", "---XMLRPC RESULT value1---",
      Dumper ($value1) );
}
```

サンプル出力

```
---XMLRPC RESULT value1 ---
{
   'vinfo' => {
      'software_version' => 'Build 4.0.90 HP RAMS',
      'appliance_version' => '0.5.24'
   },
   'numReturnedEntries' => '20',
   'network_name' => 'VOD',
   'report_time' => '20051108T20:04:47',
   'totalEntries' => '20',
   'result' => [
      {
         'numDownRoutes' => '0',
         'numActiveRoutes' => '1',
         'numNewRoutes' => '1',
         'PE' => {
            'type' => 'Originator',
            'ipaddr' => '192,168,180,180'
         },
         'deviation' => '100',
         'numBaselineRoutes' => '0',
         'vpnState' => {
            'inBaseline' => 'false',
            'down' => 'true'
         }
      }
   ]
}
```

api_vpn_routes

RPC 呼び出し：RouteAnalyzer.api_vpn_routes {password} {database name} {time} {filter}

このクエリーは、指定されたネットワークの VPN 経路リストを返します。

入力パラメータ

- **password**: クエリーに設定されたパスワード。
- **database name**: データベース階層に含まれる 1 つ以上のデータベース名 (複数の場合はスペースで区切る)。名前は、下にサブツリーを含む管理ドメイン名 (CorpNet など) か、データベース名全体 (CorpNet.BGP/AS65522/VPN など) です。
- **time**: UTC タイムゾーンの時刻を ISO 8601 形式で指定します (`20050725T21:47:35` など)。クエリーの結果は、指定された時刻のネットワークの状態に基づいて計算されます。
- **filter**: フィルターパラメータに一致するサブセットのみを出力するためのフィルター式。フィルター式の詳細は、『ユーザーガイド』の「History Navigator」の章の「式の構文」を参照してください。フィルター「any」を使用すると、結果がすべて返されます。

出力の構造

- vinfo (バージョン情報): バージョン構造体
- numReturnedEntries (返されたエントリの数): 整数
- network_name (ネットワーク名): 文字列
- report_time (レポート作成時刻): ISO 8601 形式の UTC 時刻
- totalEntries (エントリの総数): 整数
- result (結果): 以下の情報の配列
  - topology (トポロジ): トポロジ構造体
  - vpnPrefix (VPN プレフィックス):
    - labelStack (ラベルスタック): 文字列

    - prefix (プレフィックス): 文字列
- attributes (属性): BGP 属性構造体
- router (ルーター): ルーター構造体
- state (状態): 状態構造体 (基準値あり)

例

```
#!/usr/bin/perl

if (!defined($ARGV[0]) || !defined($ARGV[1])) {
   printf "usage:RouteAnalyzer.api_vpn_routes ip database\n";
   exit(0);
}

my $rexip = $ARGV[0];
my $database = $ARGV[1];
my $filter = "any";

$filter = $ARGV[2] if ($#ARGV >= 2);

use strict;
use RPC::XML::Client;
use RPC::XML 'time2iso8601';
use Date::Parse;
use Data::Dumper; $Data::Dumper::Terse = 1; $Data::Dumper::
   Indent = 1;
my $client;
my $req;
my @reqs;
my $password = 'admin';
$client = new RPC::XML::Client "http://$rexip:2000/RPC2";

my $t1 = str2time("28 Feb 2005 15:50:22 PST");

push (@reqs,
   RPC::XML::request->new('RouteAnalyzer.api_vpn_routes',
      RPC::XML::RPC_STRING($password),
      RPC::XML::RPC_STRING($database),
      RPC::XML::datetime_iso8601->new(time2iso8601($t1)),
      RPC::XML::RPC_STRING($filter)
   )
);

foreach (@reqs) {
   my $res = $client->send_request($_);
   if ($res->is_fault) {print("---XMLRPC FAULT ---"); }
   my $value1 = $res->value;
   print Dumper($value1);
}
```

サンプル出力

```
---XMLRPC RESULT value1 ---
{
   'vinfo' => {
      'software_version' => 'Build 4.0.90 HP RAMS',
      'appliance_version' => '0.5.24'
   },
   'numReturnedEntries' => '20',
   'network_name' => 'pd353',
   'report_time' => '20051027T21:40:07',
   'totalEntries' => '20',
   'result' => [
      {
         'topology' => {
            'fullName' => 'pd353.Left.BGP/AS65522/VPN',
            'protocol' => 'BGP'
         },
         'vpnPrefix' => {
            'labelStack' => '20543',
            'prefix' => '65522:700:192.168.230.230/24'
         },
         'attributes' => {
            'mpReachabilityNextHop' => '0:192.168.104.12',
            'extCommunities' => 'RT:65522:700 ',
            'origin' => 'INCOMPLETE',
            'localPref' => '100',
            'asPath' => '',
            'med' => '0'
         },
         'router' => {
            'type' => 'IBGP Peer',
            'ipaddr' => '192,168,200,200'
         },
         'state' => {
            'inBaseline' => 'false',
            'down' => 'false'
         }
      },...
   ]
}
```

api_vpn_routes_handle

RPC 呼び出し：RouteAnalyzer.api_vpn_routes_handle {password} {database} {time} {filter} {max_entries}

このクエリーは、指定されたネットワークの VPN 経路リストのハンドルを返します。

入力パラメータ

入力パラメータについては、167 ページの「api_vpn_routes」を参照してください。

出力の構造

- vinfo (バージョン情報): バージョン構造体
- numReturnedEntries (返されたエントリの数): 整数
- network_name (ネットワーク名): 文字列
- report_time (レポート作成時刻): ISO 8601 形式の UTC 時刻
- totalEntries (エントリの総数): 整数
- result (結果): 整数

例 / サンプル

27 ページの「リエントラントクエリーの使用方法」のリエントラントクエリーの説明を参照してください。

索引

D

Dumper 関数 , 18

Q

[Queries] ページ , 16

X

XML-RPC
    エラーコード , 21
    クエリー
        有効にする , 16

XML-RPC (拡張マークアップ言語用リモート
    プロシージャコール), 13

XML-RPC クエリーを有効にする , 16

え

エラーコード , 21

く

クエリー
    使用方法 , 18
    有効にする , 16
    呼び出しの説明 , 18

こ

コード、エラー , 21

せ

設定
    XML-RPC クエリーを有効にする , 16

ふ

プログラミング
    C、Java、Perl, 13