



Using External and Local Policy with Pexip Infinity

Deployment Guide

Software Version 17

Document Version 17.a

December 2017

Contents

| | |
|--|-----------|
| Using external and local policy to control Pexip Infinity behavior | 5 |
| External policy..... | 6 |
| Local policy..... | 6 |
| Whether to use external policy, local policy or both?..... | 6 |
| Configuration data request types..... | 7 |
| Policy profiles..... | 7 |
| Configuring policy profiles | 8 |
| Using the external policy server API with Pexip Infinity | 10 |
| Configuring policy profiles for external policy..... | 11 |
| Requests sent to the external policy API from Pexip Infinity | 13 |
| Request types summary..... | 13 |
| Policy server responses — general information..... | 14 |
| Dialing out from conference..... | 14 |
| Service configuration requests..... | 15 |
| Response to a service configuration request..... | 16 |
| Virtual Meeting Room / Virtual Auditorium service types response fields..... | 16 |
| Pexip Distributed Gateway service type response fields..... | 19 |
| Virtual Reception service type response fields..... | 21 |
| Test Call Service type response fields..... | 22 |
| Example service configuration data responses..... | 24 |
| Response containing service configuration data..... | 24 |
| Response instructing Pexip Infinity to reject the call..... | 24 |
| Media location requests..... | 25 |
| Media location response..... | 26 |
| Example media location data response..... | 26 |
| Participant avatar requests..... | 27 |
| Participant avatar response..... | 28 |
| Directory information requests..... | 29 |
| Directory information response..... | 29 |
| Example directory information responses..... | 30 |
| Registration alias requests..... | 30 |
| Registration alias response..... | 31 |
| Example registration alias responses..... | 31 |
| Summary of request parameters per request type..... | 33 |

| | |
|---|-----------|
| Enabling local policy | 36 |
| Configuring policy profiles for local policy..... | 36 |
| Writing local policy scripts | 37 |
| Request types summary..... | 37 |
| Writing a jinja2 script to control call behavior..... | 37 |
| Getting started in constructing a script..... | 38 |
| Supported variables..... | 39 |
| Configuration variables..... | 39 |
| Call information variables..... | 40 |
| Data manipulation filters..... | 42 |
| pex_update..... | 42 |
| pex_to_json..... | 42 |
| pex_in_subnet..... | 42 |
| pex_debug_log..... | 43 |
| Response formats..... | 43 |
| Service configuration data responses..... | 43 |
| Virtual Meeting Room / Virtual Auditorium service types response fields..... | 43 |
| Pexip Distributed Gateway service type response fields..... | 46 |
| Virtual Reception service type response fields..... | 48 |
| Test Call Service type response fields..... | 49 |
| Example service configuration data responses..... | 51 |
| Response containing service configuration data..... | 51 |
| Response instructing Pexip Infinity to reject the call..... | 52 |
| Media location data responses..... | 52 |
| Example media location data response..... | 52 |
| Dialing out from conference..... | 52 |
| Testing local policy scripts | 53 |
| Example local policy scripts | 54 |
| Minimum basic pass-through service configuration script..... | 54 |
| Basic pass-through service configuration script with call_info debug line..... | 54 |
| Minimum basic pass-through media location script..... | 55 |
| Unconditionally nominate media locations..... | 55 |
| Nominate a media location for RTMP streaming..... | 55 |
| Remove PIN based on location..... | 56 |
| Remove PIN if device is registered..... | 56 |
| Inject an ADP into every conference (with a debug line for service_config)..... | 57 |
| Reject specified User Agents..... | 57 |
| Test whether a participant's address is within a particular subnet..... | 58 |

| | |
|--|----|
| Lock a conference when the first participant connects..... | 58 |
|--|----|

Using external and local policy to control Pexip Infinity behavior

You can extend Pexip Infinity's built-in functionality by using external and/or local policy to apply bespoke call policy and routing decisions based on your own specific requirements.

For example, you might want to apply different PIN rules depending upon whether a conference participant is an on-site employee or a remote visitor, or use a specific location for media handling for certain types of calls.

External policy

Pexip Infinity's external policy API allows a vast range of call policy decisions to be taken by an external system, based on the data sources that are available to that external system.

When external policy is enabled, rather than using its own database and systems to retrieve service and participant data, Pexip Infinity Conferencing Nodes send the external policy server a service request over a RESTful API and the server should respond by returning the requested data to the Conferencing Node.

For more information about using the external policy API, see [Using the external policy server API with Pexip Infinity](#).

Local policy

Local policy allows you to manipulate service configuration and media location data (that has been provided either via the external policy API, or has been retrieved from Pexip Infinity's own database) by running a jinja2 script against that data.

For more information about using local policy, see [Enabling local policy](#).

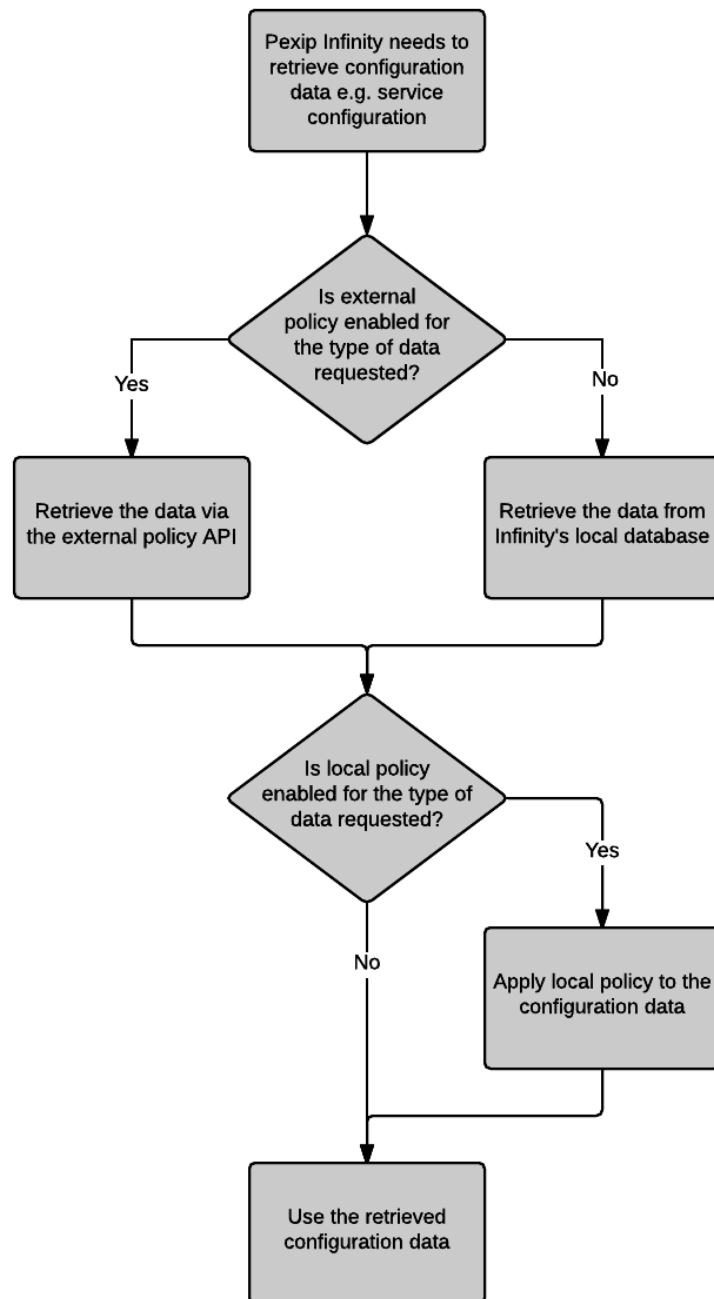
Whether to use external policy, local policy or both?

External policy provides the ultimate flexibility in implementing your own bespoke decision-making logic. However it requires more system development effort to implement and adds a dependency on a third-party system that must provide real-time responses to requests made by Pexip Infinity over the external policy API. Developing a high availability / redundant external policy server that works well in the face of network partitions can be complicated, especially if your Pexip Infinity deployment is geographically distributed.

Local policy currently has a more limited scope in what it can control than external policy, but is easier to implement and runs locally on each Conferencing Node.

External policy can be more powerful than local policy: local policy is stateless and therefore, for example, does not have access to information about any running conferences or their participants, whereas external policy can query any databases or APIs it wants, including the Pexip Infinity APIs.

You can configure Pexip Infinity to use both external and local policy depending on your requirements. When both external and local policy are enabled, external policy is applied first to retrieve the configuration data from the external system, and then local



policy is applied to that retrieved data (which can then conditionally modify that data). The flow chart (right) shows Pexip Infinity's processing logic when policy profiles are used.

Configuration data request types

The following table shows the types of configuration data that can be controlled, and by which types of policy:

| Type of data (policy requests) | Description | Controllable via external policy | Controllable via local policy |
|--------------------------------|---|----------------------------------|-------------------------------|
| Service configuration | Obtains the configuration details of a service. Pexip Infinity typically makes this request when it: <ul style="list-style-type: none"> receives an incoming call request needs to place a call to a given alias | ✓ | ✓ |
| Media location | Obtains the system location to use for media allocation. A Conferencing Node assigned to that location will handle the media for the participant. A media location request is often made after a service configuration data request. | ✓ | ✓ |
| Participant avatar | Obtains the image to display to represent a conference participant or directory contact. | ✓ | |
| Directory information | Obtains directory information — a list of device or VMR aliases and their associated names. This is used to provide phonebook information to Infinity Connect desktop clients and Infinity Connect Mobile clients that are registered to a Using External and Local Policy with Pexip Infinity Conferencing Node. | ✓ | |
| Registration alias | Used to determine whether a device alias is allowed to register to a Conferencing Node. | ✓ | |

Policy profiles

Policy profiles are used to configure how Pexip Infinity uses external policy and/or local policy to control Pexip Infinity's call policy and routing decisions.

They give you individual control over which types of policy to apply to each type of configuration data. For example, you could use both external and local policy to manipulate service configuration data, only local policy to manipulate media location data, only external policy to provide participant avatars, and no policy at all (i.e. just use the data from Pexip Infinity's own database) for directory information and registration aliases.

Each system location is configured with a policy profile and that profile is then used by all of the Conferencing Nodes in that location whenever they need to retrieve configuration data. This means that you can use the same policy profile in all locations (and thus all Conferencing Nodes), or if required you can configure many different profiles with, for example, different local policy scripts or different external policy server URIs, and then assign different policy profiles to different system locations.

See [Configuring policy profiles](#) for more information about how to set up your policy profiles.

Configuring policy profiles

Policy profiles are used to configure how Pexip Infinity uses external policy and/or local policy to control Pexip Infinity's call policy and routing decisions.

Each policy profile can be used to:

- control which types of data (e.g. service configuration, media location, participants avatars etc.) are managed via policy - either by external policy, or by local policy (where local policy is supported for that data type) or by both external and local policy
- nominate the address of an external policy server to which the external policy API requests are sent
- specify, per data type, the local policy jinja2 script to be executed against that data (currently service configuration and media location data types only).

You can configure Pexip Infinity to use both external and local policy depending on your requirements. When both external and local policy are enabled, external policy is applied first to retrieve the configuration data from the external system, and then local policy is applied to that retrieved data (which can then conditionally modify that data). See [Using external and local policy to control Pexip Infinity behavior](#) for more information.

Each system location is configured with a policy profile and that profile is then used by all of the Conferencing Nodes in that location whenever they need to retrieve configuration data. This means that you can use the same policy profile in all locations (and thus all Conferencing Nodes), or if required you can configure many different profiles with, for example, different local policy scripts or different external policy server URLs, and then assign different policy profiles to different system locations.

Note that if you are using external policy within a system location, you must ensure that each Conferencing Node in that location is able to reach the nominated policy server.

To configure policy profiles:

1. Go to **Call Control > Policy Profiles**.
2. Select **Add Policy profile** and then configure that profile. The options are:

| Option | Description |
|--|---|
| Name | The name used to refer to this policy profile in the Pexip Infinity Administrator interface. |
| Description | An optional description of the policy profile. |
| External policy server | |
| URL | The URL of the policy server to use for all external policy API requests from this profile, for example https://policy.example.com/path . You can only configure one address URL per policy server. If the request is over HTTPS, Pexip Infinity must trust the certificate presented by the policy server. |
| Username | Optional fields where you can specify the credentials required to access the external policy server. |
| Password | External policy requests support Basic Authentication and basic ASCII-encoded usernames and passwords. |
| Avatar policy | |
| Enable external avatar lookup | If enabled, requests are sent to the external policy server to fetch avatar images to represent directory contacts and conference participants. |
| Service configuration policy | |
| Enable external service configuration lookup | If enabled, requests are sent to the external policy server to fetch service configuration data (VMRs, Virtual Receptions, Pexip Distributed Gateway calls etc). |
| Apply local policy | If enabled, the service configuration retrieved from the local database or an external policy server is processed by the local policy script (which may change the service configuration or cause the call to be rejected). |

| Option | Description |
|---------------------------------------|--|
| Script | Only applies if Apply local policy is selected. Enter a jinja2 script that takes the existing service configuration (if any) and optionally modifies or overrides the service settings. |
| Media location policy | |
| Enable external media location lookup | If enabled, requests are sent to the external policy server to fetch the system location to use for media allocation. |
| Apply local policy | If enabled, the media location configuration retrieved from the local database or an external policy server is processed by the local policy script (which may change the media location configuration). |
| Script | Only applies if Apply local policy is selected. Enter a jinja2 script that takes the existing media location configuration and optionally modifies or overrides the location settings. |
| Directory | |
| Enable external directory lookup | If enabled, requests are sent to the external policy server to fetch directory information (that can be used by some Infinity Connect clients to display a phonebook). |
| Registration requests | |
| Enable external registration policy | If enabled, requests are sent to the external policy server to determine whether a device alias is allowed to register to a Conferencing Node. |

3. Select **Save**.
4. Go to **Platform Configuration > Locations**.
5. Select each location in turn and specify the **Policy profile** that the Conferencing Nodes in that location should use when making policy decisions.

Using the external policy server API with Pexip Infinity

Pexip Infinity's external policy API allows a vast range of call policy decisions to be taken by an external system, based on the data sources that are available to that external system.

When external policy is enabled, rather than using its own database and systems to retrieve service and participant data, Pexip Infinity Conferencing Nodes send the external policy server a service request over a RESTful API and the server should respond by returning the requested data to the Conferencing Node.

You must configure Pexip Infinity with the details of one or more external policy servers to which it can send policy API requests. You do this by configuring [policy profiles](#) with the addresses of one or more external policy server URLs and then associating each system location with one of those policy profiles. This means that different locations can use different policy servers for scalability and routing efficiency, if required. When a Conferencing Node needs to obtain data that is supported by external policy (such as service configuration information) it will request information from the policy server associated with the location hosting that node.

Within a policy profile you can enable or disable support for each individual request type (avatars, service configuration, media location, directory and registration requests). The requests can be subject to basic username and password authentication. For redundancy, you may also use a http load balancer in front of a pool of policy servers.

See [Requests sent to the external policy API from Pexip Infinity](#) for full information about how to use the external policy API.

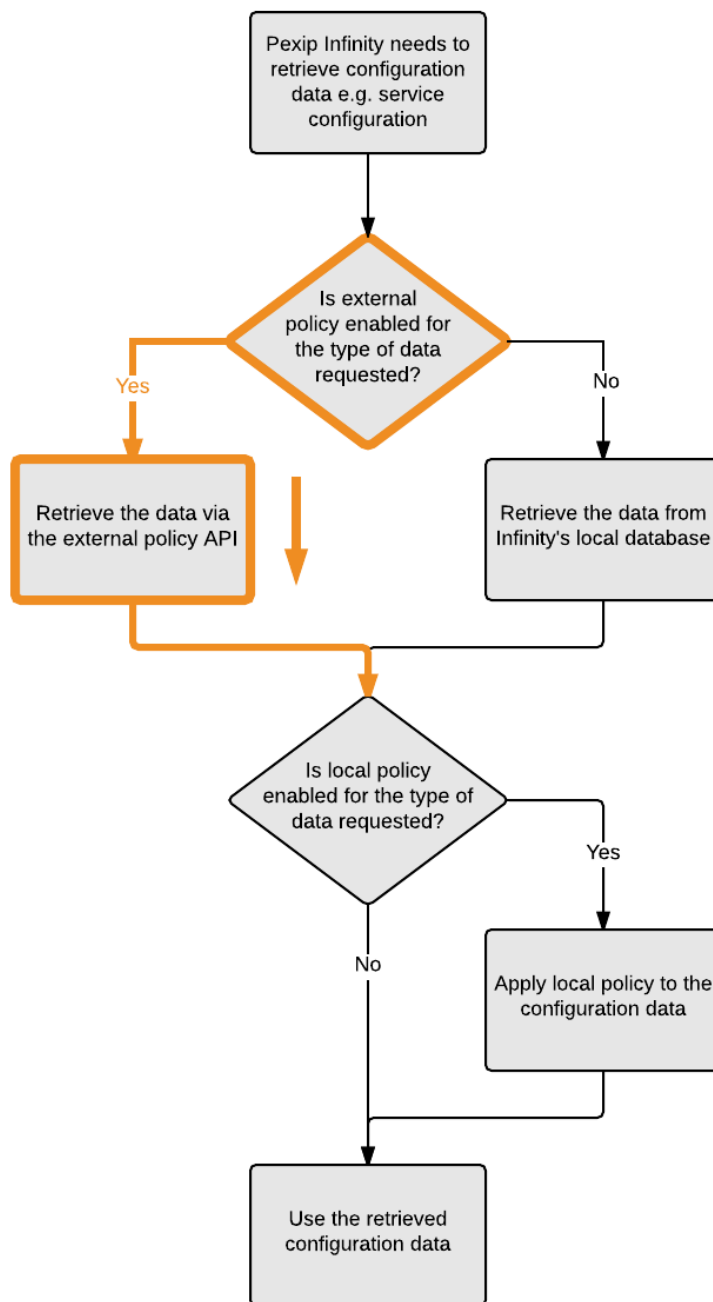
We recommend that each policy server uses the same underlying decision-making logic or database, so that consistent responses are returned. However, a policy server can apply different rules on a per-location basis, if required, as the requesting location is one of the parameters passed to the policy server in policy requests. Note that if the policy server does not return the expected service information e.g. the policy server cannot be reached, times out or returns invalid or incomplete data, Pexip Infinity will then attempt to obtain the relevant information from its own internal database (i.e. perform its standard behavior as when external policy is not in use).

Configuring policy profiles for external policy

You can configure Pexip Infinity to use both external and local policy depending on your requirements. When both external and local policy are enabled, external policy is applied first to retrieve the configuration data from the external system, and then local policy is applied to that retrieved data (which can then conditionally modify that data). The flow chart (right) shows Pexip Infinity's standard processing logic when policy profiles are used and highlights where external policy is applied.

To configure Pexip Infinity to use the external policy API:

1. Go to **Call Control > Policy Profiles**.
2. Select **Add Policy profile** and then configure that profile:
 - In the **External policy server** section, configure the URL of the policy server and any credentials required to access it.
 - Enable the external lookup options for each type of data request (avatars, service configuration, media location, directory and registration requests) that you want to submit to your external policy server, and save your changes.
3. Go to **Platform Configuration > Locations**.
4. Select each location in turn and specify the **Policy profile** that the Conferencing Nodes in that location should use when making policy decisions.



For more information on configuring policy profiles and how to combine external policy with local policy, see [Configuring policy profiles](#).

Requests sent to the external policy API from Pexip Infinity

Pexip Infinity accesses the external policy API via a GET request over HTTP or HTTPS to the appropriate external policy server URI (as configured in the policy profile).

A number of request parameters are added to the request URI according to the request type. This provides flexibility to the decision-making process within the policy server, and means, for example, that the policy server could return:

- different media location information based on the **remote_alias** or the call **protocol**
- a different avatar for the same participant based on the originally dialed **local_alias** or the **service_name**
- a different set of directory contacts depending on the **registered_alias** making the request.

This topic contains a [summary](#) of the supported requests and [response](#) guidelines. Each request type is then explained in more detail (see [service configuration](#), [media location](#), [participant avatar](#), [directory information](#) and [registration alias](#)) including the parameters sent to the policy server for that request, the expected response and some examples. Finally, there is a [summary matrix](#) showing which parameters are contained in each policy request type.

Request types summary

The following table shows the different API request types that a Conferencing Node running version 17 can send to an external policy server.

| Policy request type | Description and request URI |
|---------------------------------------|--|
| Service configuration | Obtains the configuration details of a service. Pexip Infinity typically makes this request when it: <ul style="list-style-type: none"> • receives an incoming call request • needs to place a call to a given alias Request URI: <policy_server_uri>/policy/v1/service/configuration |
| Media location | Obtains the system location to use for media allocation. A Conferencing Node assigned to that location will handle the media for the participant. A media location request is often made after a service configuration data request. Request URI: <policy_server_uri>/policy/v1/participant/location |
| Participant avatar | Obtains the image to display to represent a conference participant or directory contact. Request URI: <policy_server_uri>/policy/v1/participant/avatar/<alias> where <alias>* is the alias of the participant/contact whose avatar is required. |
| Directory information | Obtains directory information — a list of device or VMR aliases and their associated names. This is used to provide phonebook information to Infinity Connect desktop clients and Infinity Connect Mobile clients that are registered to a Using External and Local Policy with Pexip Infinity Conferencing Node. Request URI: <policy_server_uri>/policy/v1/registrations |
| Registration alias | Used to determine whether a device alias is allowed to register to a Conferencing Node. Request URI: <policy_server_uri>/policy/v1/registrations/<alias> where <alias>* is the alias of the device that is making the registration request. |

* The alias may include scheme information, for example sip:alice@example.com. The policy server must be able to parse the received alias in an appropriate manner.

Note that:

- You can configure whether specific request types are sent or not, on a per policy profile basis, via **Call Control > Policy Profiles**. If a request type is disabled, Pexip Infinity will fall back to its own default behavior for that request type.
- Pexip Infinity has a non-configurable timeout of 5 seconds for each attempt it makes to contact a policy server.

- Most policy requests are sent from the Conferencing Node that is handling the call signaling or has received the registration request. However, requests are sent from the Conferencing Node that is handling the call media in the following situations:
 - It is a participant avatar request.
 - The participant is in a Virtual Reception (note that when the endpoint initially calls the Virtual Reception alias, the requests will come from the Conferencing Node handling the signaling — as usual, but after being placed into the Virtual Reception all subsequent requests i.e. when the participant enters the number of the VMR they want to join, are sent from the Conferencing Node that is handling the Virtual Reception call media).
- You may see multiple service configuration and media location requests when handling Infinity Connect participants.

Policy server responses — general information

The policy server response to a request should be a 200 OK message. The response header must include the Content-Type, and the message body must include the requested content.

The response for all requests, except for participant avatar requests, must return a **Content-Type** of **application/json**. The policy server can return an error code e.g. 404 if it wants Pexip Infinity to fall back to its own default behavior for a specific request.

The response to a request takes the basic format:

```
{
  "status" : "success",
  "action" : "reject|continue",
  "result" : { <data> },
  "<other_keys>" : "<other_values>"
}
```

where:

- if "status" is anything other than "success", the response is deemed to have failed and Pexip Infinity will fall back to its default behavior (which is to attempt to retrieve the relevant data from its own internal database), unless the "action" field is supported in the response for that request type, in which case the "action" field instructs Pexip Infinity how to proceed.
- "action" is an optional value that may be included in responses to service configuration and registration alias requests. When present, it instructs Pexip Infinity how to proceed in failure scenarios:
 - "reject" instructs Pexip Infinity to reject the request — for a service configuration request this would mean to reject the call immediately (i.e. return "conference not found"), and for a registration request it would mean to reject the registration.
 - "continue" (or any other value except "reject") instructs Pexip Infinity to fall back to its default behavior (which is to attempt to retrieve the service configuration or device alias data from its own internal database).
- "result" is a JSON object of key value pairs as appropriate for the request type. The specific requirements of what must be included in the result is included below in the descriptions of each request type. The fields in the JSON object can be supplied in any order.
- "<other_keys>" and "<other_values>" can be zero, one or more optional key value pairs. If included, they do not affect how Pexip Infinity processes the response but they will be included in any associated support log messages. They could, for example, indicate the version number of the software running on the policy server, or contain "reason" information for failure responses.

Note that responses to participant avatar requests have different requirements (see [Participant avatar response](#) for details).

Dialing out from conference

Pexip Infinity makes a service configuration request if it dials out from a conference in order to invite a participant to join (where the `call_direction` parameter will be `dial_out`). In these cases, the response to the service configuration request must match the existing service data (i.e. the same `name`, `service_type` and so on).

When dialing out, the only configuration you can control via policy is the media location — you can do this in the response to the media location request that follows the service configuration request.

Service configuration requests

Obtains the configuration details of a service. Pexip Infinity typically makes this request when it:

- receives an incoming call request
- needs to place a call to a given alias

Request URI: `<policy_server_uri>/policy/v1/service/configuration`

Pexip Infinity includes the following fields in a service configuration request:

| Parameter | Description |
|----------------------|--|
| bandwidth | The maximum requested bandwidth for the call. |
| call_direction | The direction of the call that triggered the request. Values can be: <ul style="list-style-type: none"> • "dial_in": calls in to Pexip Infinity • "dial_out": calls dialed out from Pexip Infinity • "non_dial": when policy is triggered by other requests that are not related to incoming or outgoing call setup, such as when an H.323 LRQ or a SIP SUBSCRIBE or OPTIONS request is received. |
| local_alias | In the context of service configuration requests, this is the incoming alias (typically the alias that the endpoint has dialed). This is the primary item of information that the policy server will use to return appropriate service configuration data. |
| location | The name of the Pexip Infinity system location associated with the Conferencing Node making the request/notification. |
| ms-subnet † | The sender's subnet address. |
| node_ip | The IP address of the Conferencing Node making the request. |
| p_Asserted-Identity† | The authenticated identity of the user sending the SIP message. |
| protocol | The call protocol. Values: "api", "webrtc", "sip", "rtmp", "h323" or "mssip". (Note that the protocol is always reported as "api" when an Infinity Connect client dials in to Pexip Infinity.) |
| pseudo_version_id | The Pexip Infinity software build number. |
| remote_address | The IP address of the remote participant. |
| remote_alias | The name of the user or the registered alias of the endpoint. The <code>remote_alias</code> may include scheme information, for example <code>sip:alice@example.com</code> . |
| remote_display_name | The display name of the remote participant. |
| remote_port | The IP port of the remote participant. |
| registered | Whether the remote participant is registered or not. Values: "True" or "False". |
| trigger | The trigger for the policy request. Values: "web", "web_avatar_fetch", "invite", "options", "subscribe", "setup", "arq", "lrq" or "unspecified". |
| vendor | System details about the remote participant, such as manufacturer name and version number for hard endpoints, or browser and operating system details for soft clients. |
| version_id | The Pexip Infinity software version number. |

† only included if the request was triggered by a SIP message, and the parameter is included as a field in the SIP message header

This example shows a GET request for service configuration data when `alice@example.com` has dialed `meet.bob` from a SIP endpoint:

```
GET /example.com/policy/v1/service/configuration?protocol=sip&node_ip=10.44.99.2&registered=False&remote_address=10.44.75.249&version_id=16&bandwidth=0&pseudo_version_id=36402.0.0&vendor=TANDBERG/518 (TC6.0.1.65adebe)&local_alias=meet.bob&remote_port=58435&call_direction=dial_in&remote_alias=sip:alice@example.com&remote_display_name=Alice&trigger=invite&location=London
```

This example shows a call from an mssip (Skype for Business / Lync) endpoint:

```
GET /example.com/policy/v1/service/configuration?P-Asserted-Identity="Alice"<sip:alice@example.com>&protocol=mssip&node_ip=10.47.2.43&registered=False&remote_address=10.47.2.20&version_id=16&bandwidth=0&pseudo_version_id=36402.0.0&vendor=UCCAPI/16.0.7967.5277 OC/16.0.7967.2139 (Skype for Business)&local_alias=sip:meet.alice@example.com&remote_port=63726&call_direction=dial_in&remote_alias=sip:alice@example.com&remote_display_name=Alice&trigger=invite&location=London
```

Response to a service configuration request

The response to a service configuration request takes the basic format:

```
{
  "status" : "success",
  "action" : "reject|continue",
  "result" : { <service_configuration_data> }
}
```

where:

- "action" is an optional value that, when included, instructs Pexip Infinity how to proceed in failure scenarios:
 - "reject" instructs Pexip Infinity to reject the call immediately (i.e. return "conference not found")
 - "continue" instructs Pexip Infinity to fall back to its default behavior (which is to attempt to retrieve the service configuration data from its own internal database)
- "result" is a JSON object of multiple key value pairs that describes the service. Some data fields are required, and some are optional. The fields expected by Pexip Infinity in the response depend upon the returned `service_type` — either "conference" or "lecture" for a Virtual Meeting Room or Virtual Auditorium, "gateway" for a Pexip Distributed Gateway call, "two_stage_dialing" for a Virtual Reception, or "test_call" for a Test Call Service.
- the "action" field is ignored if "status" is "success" and "result" contains valid data (and it is a 200 OK message).

The fields expected in the response for each service type are described below:

Virtual Meeting Room / Virtual Auditorium service types response fields

Pexip Infinity expects the following fields to be returned for a `service_type` of "conference" (VMR) or "lecture" (Virtual Auditorium):

| Field name | Required | Type | Description |
|--------------|----------|--------|---|
| service_type | Yes | String | The type of service, in this case: <ul style="list-style-type: none">• "conference": a Virtual Meeting Room, or• "lecture": a Virtual Auditorium |

| Field name | Required | Type | Description |
|--|----------|---------|--|
| name | Yes | String | The name of the service. You could, for example, use the <code>local_alias</code> received in the configuration request. Pexip Infinity will subsequently use this name — as the <code>service_name</code> parameter — in subsequent requests. |
| description | No | String | A description of the service. |
| service_tag | Yes | String | A unique identifier used to track usage of this service. |
| pin | No | String | Host PIN — the secure access code for Host participants. |
| allow_guests | No | Boolean | Whether to distinguish between Host and Guest participants: <ul style="list-style-type: none"> • <code>true</code>: the conference has two types of participants: Hosts and Guests. The <code>pin</code> to be used by Hosts must be specified. A <code>guest_pin</code> can optionally be specified; if a <code>guest_pin</code> is not specified, Guests can join without a PIN. • <code>false</code>: all participants have Host privileges Default: <code>false</code> |
| guest_pin | No | String | Guest PIN — the secure access code for Guest participants. |
| guests_can_present | No | Boolean | Controls whether the Guests in the conference are allowed to present content. <ul style="list-style-type: none"> • <code>true</code>: Guests and Hosts can present into the conference • <code>false</code>: only Hosts can present Default: <code>true</code> |
| max_callrate_in | No | Integer | The maximum media bandwidth in bps that Pexip Infinity will receive from each individual participant dialed in to the service. Range 128000 to 4096000 bps. |
| max_callrate_out | No | Integer | The maximum media bandwidth in bps that Pexip Infinity will send to each individual participant dialed in to the service. Range 128000 to 4096000 bps. |
| participant_limit | No | Integer | The maximum number of participants allowed to join the service. |
| ivr_theme_name | No | String | The name of the theme to use with the service. If no theme is specified, the default Pexip theme is used. |
| automatic_participants | No | List | A list of participants to dial automatically when the conference starts. Each participant in the list contains a number of fields as described in Automatically dialed participants (ADP) fields . |
| view (applies to service_type of "conference" only) | No | String | The layout seen by all participants: <ul style="list-style-type: none"> • <code>"one_main_zero_pips"</code>: full-screen main speaker only • <code>"one_main_seven_pips"</code>: large main speaker and up to 7 other participants • <code>"one_main_twentyone_pips"</code>: main speaker and up to 21 other participants • <code>"two_mains_twentyone_pips"</code>: two main speakers and up to 21 other participants Default: <code>"one_main_seven_pips"</code> |

| Field name | Required | Type | Description |
|---|----------|---------|---|
| host_view (applies to service_type of "lecture" only) | No | String | <p>The layout seen by Hosts:</p> <ul style="list-style-type: none"> "one_main_zero_pips": full-screen main speaker only "one_main_seven_pips": large main speaker and up to 7 other participants "one_main_twentyone_pips": main speaker and up to 21 other participants "two_mains_twentyone_pips": two main speakers and up to 21 other participants <p>Default: "one_main_seven_pips"</p> |
| guest_view (applies to service_type of "lecture" only) | No | String | <p>The layout seen by Guests:</p> <ul style="list-style-type: none"> "one_main_zero_pips": full-screen main speaker only "one_main_seven_pips": large main speaker and up to 7 other participants "one_main_twentyone_pips": main speaker and up to 21 other participants "two_mains_twentyone_pips": two main speakers and up to 21 other participants <p>Default: "one_main_seven_pips"</p> <p>Only Hosts are displayed. Guests can hear but not see any of the other Guests.</p> |
| enable_overlay_text | No | Boolean | <p>If participant name overlays are enabled, the display name or alias of the participants (the current main speaker and thumbnails) is shown in a text overlay along the bottom of their video image.</p> <ul style="list-style-type: none"> true: participant names are shown false: participant names are not shown <p>Default: false</p> |
| locked | No | Boolean | <p>Whether to lock the conference on creation:</p> <ul style="list-style-type: none"> true: the conference will be locked on creation false: the conference will not be locked <p>Note that this field has no effect on the conference if it is already running.</p> <p>Default: false</p> |
| call_type | No | String | <p>The call capability of the conference. It can be limited to:</p> <ul style="list-style-type: none"> "video": main video plus presentation "video-only": main video only "audio": audio-only <p>Default: "video"</p> |
| enable_chat | No | String | <p>Whether chat messaging is enabled for the conference:</p> <ul style="list-style-type: none"> "default": as per the global configuration setting "yes": chat is enabled "no": chat is disabled <p>Default: "default"</p> |
| primary_owner_email_address | No | String | The email address of the owner of the VMR. |

| Field name | Required | Type | Description |
|--|----------|---------|--|
| force_presenter_into_main (applies to service_type of "lecture" only) | No | Boolean | Controls whether the Host who is presenting is locked into the main video position: <ul style="list-style-type: none"> true: the Host sending the presentation stream will always hold the main video position false: the main video position is voice-switched Default: false |
| mute_all_guests (applies to service_type of "lecture" only) | No | Boolean | Controls whether to mute guests when they first join the conference: <ul style="list-style-type: none"> true: mute Guests when they first join the conference false: do not mute Guests when they first join the conference Default: false |

Pexip Distributed Gateway service type response fields

Pexip Infinity expects the following fields to be returned for a **service_type** of "gateway":

| Field name | Required | Type | Description |
|--------------------|----------|---------|--|
| service_type | Yes | String | The type of service, in this case: <ul style="list-style-type: none"> "gateway": a Pexip Distributed Gateway call |
| name | Yes | String | The name of the service. Ensure that all gateway service instances have a unique name to avoid conflicts between calls. Pexip Infinity will subsequently use this name — as the service_name parameter — in subsequent requests. |
| description | No | String | A description of the service. |
| service_tag | Yes | String | A unique identifier used to track usage of this service. |
| max_callrate_in | No | Integer | The maximum media bandwidth in bps that Pexip Infinity will receive from each individual participant dialed in to the service. Range 128000 to 4096000 bps. |
| max_callrate_out | No | Integer | The maximum media bandwidth in bps that Pexip Infinity will send to each individual participant dialed in to the service. Range 128000 to 4096000 bps. |
| ivr_theme_name | No | String | The name of the theme to use with the service. If no theme is specified, the default Pexip theme is used. |
| remote_alias | Yes | String | The alias of the endpoint to call. The alias can include scheme information, for example sip:alice@example.com , but the call will always be made using the specified outgoing_protocol . |
| local_alias | Yes | String | The calling or "from" alias. This is the alias that the recipient would use to return the call. |
| local_display_name | No | String | The display name of the calling alias. |

| Field name | Required | Type | Description |
|------------------------|----------|--------|--|
| outgoing_protocol | Yes | String | The protocol to use to place the outgoing call: <ul style="list-style-type: none"> "h323": an H.323 call; you can also optionally nominate an <code>h323_gatekeeper_name</code> "sip": a SIP call; you can also optionally nominate a <code>sip_proxy_name</code> "mssip": a Microsoft Skype for Business / Lync call; you can also optionally nominate an <code>mssip_proxy_name</code> and a <code>turn_server_name</code> "rtmp": uses the RTMP protocol; typically this is used for content streaming |
| outgoing_location_name | No | String | The name* of the location of a Conferencing Node from which to place the call. |
| h323_gatekeeper_name | No | String | The name* of the H.323 gatekeeper to use to place the outgoing call. DNS is used if no gatekeeper is specified. |
| sip_proxy_name | No | String | The name* of the SIP proxy to use to place the outgoing call. DNS is used if no proxy is specified. |
| mssip_proxy_name | No | String | The name* of the Skype for Business / Lync server to use to place the outgoing call. DNS is used if no server is specified. |
| turn_server_name | No | String | The name* of the TURN server to offer to external SIP and WebRTC endpoints that are ICE-enabled (such as Skype for Business / Lync clients and Infinity Connect WebRTC clients). |
| stun_server_name | No | String | The name* of the STUN server to use when placing calls to external SIP and WebRTC endpoints that are ICE-enabled (such as Skype for Business / Lync clients and Infinity Connect WebRTC clients). |
| call_type | No | String | The call capability of the outbound call: <ul style="list-style-type: none"> "video": main video plus presentation "video-only": main video only "audio": audio-only "auto": match the capability of the outbound call to that of the inbound call Default: "video" |
| called_device_type | No | String | The device or system to which to route the call: <ul style="list-style-type: none"> "external": route the call to a matching registered device if it is currently registered, otherwise attempt to route the call via an external system "registration": route the call to a matching registered device only (providing it is currently registered) "mssip_conference_id": route the call via a Skype for Business / Lync server to a SfB/Lync meeting where the <code>remote_alias</code> is a SfB/Lync meeting Conference ID "mssip_server": route the call via a Skype for Business / Lync server to a SfB/Lync client or meeting Default: "external" |

* The returned "name" must match the name of the location, H.323 gatekeeper, SIP proxy etc. configured within Pexip Infinity.

Virtual Reception service type response fields

Pexip Infinity expects the following fields to be returned for a **service_type** of "two_stage_dialing" (Virtual Reception):

| Field name | Required | Type | Description |
|----------------------|----------|---------|--|
| service_type | Yes | String | The type of service, in this case: <ul style="list-style-type: none"> "two_stage_dialing": a Virtual Reception |
| name | Yes | String | The name of the service. You could, for example, use the local_alias received in the configuration request. Pexip Infinity will subsequently use this name — as the service_name parameter — in subsequent requests. |
| description | No | String | A description of the service. |
| service_tag | Yes | String | A unique identifier used to track usage of this service. |
| max_callrate_in | No | Integer | The maximum media bandwidth in bps that Pexip Infinity will receive from each individual participant dialed in to the service. Range 128000 to 4096000 bps. |
| max_callrate_out | No | Integer | The maximum media bandwidth in bps that Pexip Infinity will send to each individual participant dialed in to the service. Range 128000 to 4096000 bps. |
| ivr_theme_name | No | String | The name of the theme to use with the service. If no theme is specified, the default Pexip theme is used. |
| call_type | No | String | The call capability of the reception. It can be limited to: <ul style="list-style-type: none"> "video": main video plus presentation "video-only": main video only "audio": audio-only Default: "video" |
| mssip_proxy_name | No | String | The name* of the Skype for Business / Lync server to use to place the outgoing call. DNS is used if no server is specified. |
| system_location_name | No | String | This is an optional field used in conjunction with the mssip_proxy_name setting. If specified, a Conferencing Node in this system location will perform the SfB/Lync Conference ID lookup on the SfB/Lync server. If a location is not specified, the IVR ingress node will perform the lookup. |
| match_string | No | String | An optional regular expression used to match against the alias entered by the caller into the Virtual Reception. If the entered alias does not match the expression, the Virtual Reception will not route the call. |
| replace_string | No | String | An optional regular expression used to transform the alias entered by the caller into the Virtual Reception. (Only applies if a regex match string is also configured and the entered alias matches that regex.) Leave this field blank if you do not want to change the alias entered by the caller. |

Test Call Service type response fields

Pexip Infinity expects the following fields to be returned for a **service_type** of "test_call" (Test Call Service):

| Field name | Required | Type | Description |
|-----------------|----------|---------|---|
| service_type | Yes | String | The type of service, in this case: <ul style="list-style-type: none"> "test_call": a Test Call Service |
| name | Yes | String | The name of the service. You could, for example, use the local_alias received in the configuration request. Pexip Infinity will subsequently use this name — as the service_name parameter — in subsequent requests. |
| description | No | String | A description of the service. |
| service_tag | Yes | String | A unique identifier used to track usage of this service. |
| max_callrate_in | No | Integer | The maximum media bandwidth in bps that Pexip Infinity will receive from each individual participant dialed in to the service. Range 128000 to 4096000 bps. |
| ivr_theme_name | No | String | The name of the theme to use with the service. If no theme is specified, the default Pexip theme is used. |
| call_type | No | String | The call capability of the service. It can be limited to: <ul style="list-style-type: none"> "video": main video plus presentation "video-only": main video only "audio": audio-only Default: "video" |

Automatically dialed participants (ADP) fields

The **automatic_participants** list field, which may be included in the service configuration response for a **service_type** of "conference" (VMR) or "lecture" (Virtual Auditorium), contains the following fields per participant:

| Field name | Required | Type | Description |
|---------------------|----------|--------|---|
| remote_alias | Yes | String | The alias of the endpoint to call. |
| remote_display_name | No | String | An optional friendly name for this participant. This may be used instead of the participant's alias in participant lists and as a text overlay in some layout configurations. |
| local_alias | Yes | String | The calling or "from" alias. This is the alias that the recipient would use to return the call. |
| local_display_name | No | String | The display name of the calling or "from" alias. |
| routing | No | String | Specifies how to route the call: <ul style="list-style-type: none"> "manual": uses the requested protocol and the defaults for the specified system_location_name. "routing_rule": routes the call according to the configured Call Routing Rules— this means that the dialed alias must match an outgoing Call Routing Rule for the call to be placed (using the protocols and call control systems etc. as configured for that rule). Default: "manual" |

| Field name | Required | Type | Description |
|-----------------------|----------|---------|---|
| protocol | Yes | String | <p>The protocol to use to place the outgoing call:</p> <ul style="list-style-type: none"> "h323": an H.323 call "sip": a SIP call "mssip": a Microsoft Skype for Business / Lync call "rtmp": uses the RTMP protocol; typically this is used for content streaming <p>Calls are routed to externally-located participants based on the configuration of the system location used to place the call.</p> |
| role | Yes | String | <p>The level of privileges the participant has in the conference:</p> <ul style="list-style-type: none"> "chair": the participant has Host privileges "guest": the participant has Guest privileges |
| dtmf_sequence | No | String | An optional DTMF sequence to be transmitted after the call to the dialed participant starts. |
| system_location_name | No | String | The location of the Conferencing Node from which to place the call. |
| streaming | No | Boolean | <p>Identifies the dialed participant as a streaming or recording device.</p> <ul style="list-style-type: none"> true: the participant is a streaming or recording device false: the participant is not a streaming or recording device <p>Default: false</p> |
| keep_conference_alive | No | String | <p>Determines whether the conference continues when all other non-ADP participants have disconnected:</p> <ul style="list-style-type: none"> "keep_conference_alive": the conference continues to run until this participant disconnects (applies to Hosts only). "keep_conference_alive_if_multiple": the conference continues to run as long as there are two or more "keep_conference_alive_if_multiple" participants and at least one of them is a Host. "keep_conference_alive_never": the conference terminates automatically if this is the only remaining participant. <p>Default: "keep_conference_alive_if_multiple"</p> <p>For more information, see https://docs.pexip.com/admin/automatically_terminate.htm.</p> |
| call_type | No | String | <p>The call capability of the participant. It can be limited to:</p> <ul style="list-style-type: none"> "video": main video plus presentation "video-only": main video only "audio": audio-only <p>Default: "video"</p> |
| presentation_url | No | String | This additional parameter can be specified for RTMP calls to send the presentation stream to a separate RTMP destination. |

Example service configuration data responses

Response containing service configuration data

This is an example response of service configuration data for a "conference" service type:

```
{
  "status" : "success",
  "action" : "continue",
  "result" : {
    "service_type" : "conference",
    "name" : "Alice Jones",
    "service_tag" : "abcd1234",
    "description" : "Alice Jones personal VMR",
    "pin" : "1234",
    "allow_guests" : true,
    "guest_pin" : "5678",
    "view" : "one_main_zero_pips",
    "enable_overlay_text" : true,
    "automatic_participants" :
    [
      {
        "remote_alias" : "sip:alice@example.com",
        "remote_display_name" : "Alice",
        "local_alias" : "meet.alice@example.com",
        "local_display_name" : "Alice's VMR",
        "protocol" : "sip",
        "role" : "chair",
        "system_location_name" : "London"
      },
      {
        "remote_alias" : "rtmp://example.com/live/alice_vmr",
        "local_alias" : "meet.alice@example.com",
        "local_display_name" : "Alice's VMR",
        "protocol" : "rtmp",
        "role" : "guest",
        "streaming" : true
      }
    ]
  },
  "xyz_version" : "1.2"
}
```

Response instructing Pexip Infinity to reject the call

This is an example response that tells Pexip Infinity to reject a call. This could be sent in a 200 OK message, or in a 404 response:

```
{
  "status" : "success",
  "action" : "reject"
}
```

Note that Pexip Infinity will also reject the call if "status" is set to a different value (such as "failure") or if "status" is omitted entirely.

Media location requests

Obtains the system location to use for media allocation. A Conferencing Node assigned to that location will handle the media for the participant. A media location request is often made after a service configuration data request.

Request URI: `<policy_server_uri>/policy/v1/participant/location`

Pexip Infinity includes the following fields in a media location request:

| Parameter | Description |
|----------------------|--|
| bandwidth | The maximum requested bandwidth for the call. |
| call_direction | The direction of the call that triggered the request. Values can be: <ul style="list-style-type: none"> "dial_in": calls in to Pexip Infinity "dial_out": calls dialed out from Pexip Infinity "non_dial": when policy is triggered by other requests that are not related to incoming or outgoing call setup, such as when an H.323 LRQ or a SIP SUBSCRIBE or OPTIONS request is received. |
| local_alias | For participant requests (media location and participant avatar) this contains the originally-dialed incoming alias for the associated service. |
| location | The name of the Pexip Infinity system location associated with the Conferencing Node making the request/notification. |
| ms-subnet † | The sender's subnet address. |
| node_ip | The IP address of the Conferencing Node making the request. |
| p_Asserted-Identity† | The authenticated identity of the user sending the SIP message. |
| protocol | The call protocol. Values: "api", "webrtc", "sip", "rtmp", "h323" or "mssip". (Note that the protocol is always reported as "api" when an Infinity Connect client dials in to Pexip Infinity.) |
| pseudo_version_id | The Pexip Infinity software build number. |
| remote_address | The IP address of the remote participant. |
| remote_alias | The name of the user or the registered alias of the endpoint. The <code>remote_alias</code> may include scheme information, for example <code>sip:alice@example.com</code> . |
| remote_display_name | The display name of the remote participant. |
| remote_port | The IP port of the remote participant. |
| registered | Whether the remote participant is registered or not. Values: "True" or "False". |
| service_name | The service name. This will match the name field returned by the policy server from the original service configuration request. |
| service_tag | The service tag associated with the <code>service_name</code> parameter. |
| trigger | The trigger for the policy request. Values: "web", "web_avatar_fetch", "invite", "options", "subscribe", "setup", "arq", "lrq" or "unspecified". |

| Parameter | Description |
|---|--|
| unique_service_name | The unique name used by Pexip Infinity to identify the service: <ul style="list-style-type: none"> For "gateway" services this is the matching rule name followed by a unique identifier (so as to distinguish between separate calls that match the same rule). For all other services, this is the same as the service_name parameter. |
| vendor | System details about the remote participant, such as manufacturer name and version number for hard endpoints, or browser and operating system details for soft clients. |
| version_id | The Pexip Infinity software version number. |
| † only included if the request was triggered by a SIP message, and the parameter is included as a field in the SIP message header | |

This example shows a GET request for the media location to use for alice@example.com:

```
GET /example.com/policy/v1/participant/location?protocol=sip&node_ip=10.44.99.2&service_name=meet.bob&registered=False&remote_address=10.44.75.250&version_id=16&service_tag=&bandwidth=0&pseudo_version_id=36402.0.0&vendor=TANDBERG/518 (TC6.0.1.65adebe) &unique_service_name=meet.bob&local_alias=meet.bob&remote_port=58426&call_direction=dial_in&remote_alias=sip:alice@example.com&remote_display_name=Alice&trigger=invite&location=London
```

Media location response

The response to a media location request takes the format:

```
{
  "status" : "success",
  "result" : { <location_data> }
}
```

where "result" is a JSON object of key value pairs that describes the media location to use. The fields that may be included are:

| Field name | Required | Description |
|-----------------------------|----------|---|
| location | Yes | The name* of the principal location to use for media allocation. A Conferencing Node assigned to that location will handle the media for the participant. |
| primary_overflow_location | No | The name* of the system location to handle the media if the principal location has reached its capacity. |
| secondary_overflow_location | No | The name* of the system location to handle the media if both the principal location and the primary overflow location have reached their capacity. |

* The returned "name" must match the name of a location configured within Pexip Infinity. If any of the location names included in the response do not match a configured location within Pexip Infinity, the entire response is deemed to have failed and Pexip Infinity will fall back to its own default behavior for that request.

Example media location data response

This is an example response containing media location data:

```
{
  "status" : "success",
  "result" : {
    "location" : "London",
    "primary_overflow_location" : "Oslo"
  }
}
```

Participant avatar requests

Obtains the image to display to represent a conference participant or directory contact.

Request URI: `<policy_server_uri>/policy/v1/participant/avatar/<alias>` where `<alias>` is the alias of the participant/contact whose avatar is required.

Pexip Infinity includes the following fields in a participant avatar request:

| Parameter | Description |
|----------------------|--|
| bandwidth | The maximum requested bandwidth for the call. |
| call_direction | The direction of the call that triggered the request. Values can be: <ul style="list-style-type: none"> "dial_in": calls in to Pexip Infinity "dial_out": calls dialed out from Pexip Infinity "non_dial": when policy is triggered by other requests that are not related to incoming or outgoing call setup, such as when an H.323 LRQ or a SIP SUBSCRIBE or OPTIONS request is received. |
| height * | The required height in pixels of the image to be returned. |
| local_alias | For participant requests (media location and participant avatar) this contains the originally-dialed incoming alias for the associated service. |
| location * | The name of the Pexip Infinity system location associated with the Conferencing Node making the request/notification. |
| ms-subnet † | The sender's subnet address. |
| node_ip * | The IP address of the Conferencing Node making the request. |
| p_Asserted-Identity† | The authenticated identity of the user sending the SIP message. |
| protocol | The call protocol. Values: "api", "webrtc", "sip", "rtmp", "h323" or "mssip". (Note that the protocol is always reported as "api" when an Infinity Connect client dials in to Pexip Infinity.) |
| pseudo_version_id * | The Pexip Infinity software build number. |
| remote_address | The IP address of the remote participant. |
| remote_alias | The name of the user or the registered alias of the endpoint. The <code>remote_alias</code> may include scheme information, for example <code>sip:alice@example.com</code> . |
| remote_display_name | The display name of the remote participant. |
| remote_port | The IP port of the remote participant. |
| registered | Whether the remote participant is registered or not. Values: "True" or "False". |
| role | The role associated with the <code>remote_alias</code> . Values: "chair" (for Host participants), "guest" or "unknown" (a participant who is at the PIN entry screen). |

| Parameter | Description |
|--|--|
| service_name | The service name. This will match the name field returned by the policy server from the original service configuration request. |
| service_tag | The service tag associated with the service_name parameter. |
| trigger | The trigger for the policy request. Values: "web", "web_avatar_fetch", "invite", "options", "subscribe", "setup", "arq", "lrq" or "unspecified". |
| unique_service_name | The unique name used by Pexip Infinity to identify the service: <ul style="list-style-type: none"> For "gateway" services this is the matching rule name followed by a unique identifier (so as to distinguish between separate calls that match the same rule). For all other services, this is the same as the service_name parameter. |
| vendor | System details about the remote participant, such as manufacturer name and version number for hard endpoints, or browser and operating system details for soft clients. |
| version_id * | The Pexip Infinity software version number. |
| width * | The required width in pixels of the image to be returned. |
| * these are the only parameters included in a participant avatar request when Pexip Infinity is retrieving directory information (as opposed to service participant-related information where all parameters are included) | |
| † only included if the request was triggered by a SIP message, and the parameter is included as a field in the SIP message header | |

This example shows a GET request for a participant avatar for the alias "alice" who is in the conference "meet.bob", where the participant device is an unregistered Infinity Connect client:

```
GET /example.com/policy/v1/participant/avatar/alice?protocol=webrtc&node_ip=10.44.99.2&service_name=meet.bob&registered=False&remote_address=10.44.75.250&version_id=16&service_tag=&bandwidth=0&pseudo_version_id=36402.0.0&vendor=Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/55.0.2883.87 Safari/537.36&height=100&unique_service_name=meet.bob&local_alias=meet.bob&remote_port=58426&call_direction=dial_in&remote_alias=alice&remote_display_name=Alice&width=100&trigger=invite&role=chair&location=London
```

This example shows a GET request for a participant avatar for the alias "alice" that has been triggered in response to a previous directory information request:

```
GET /example.com/policy/v1/participant/avatar/alice?node_ip=10.47.2.46&registered=True&version_id=16&height=40&width=40&location=London&pseudo_version_id=36402.0.0
```

Participant avatar response

The response for an avatar request must return a **Content-Type** of **image/jpeg**.

All JPEG images must use the RGB or RGBA color space (CMYK is not supported), and be of the requested size (width, height).

If a valid JPEG image is not returned or the image is the wrong size, Pexip Infinity will return a 404 Not Found response to the caller (i.e. the Infinity Connect client) which will then use its standard placeholder participant image.

Directory information requests

Obtains directory information — a list of device or VMR aliases and their associated names. This is used to provide phonebook information to Infinity Connect desktop clients and Infinity Connect Mobile clients that are registered to a Using External and Local Policy with Pexip Infinity Conferencing Node.

Request URI: `<policy_server_uri>/policy/v1/registrations`

Pexip Infinity includes the following fields in a directory information request:

| Parameter | Description |
|--------------------|---|
| limit | The maximum number of results to return (currently this is always set to 10). |
| location | The name of the Pexip Infinity system location associated with the Conferencing Node making the request/notification. |
| node_ip | The IP address of the Conferencing Node making the request. |
| pseudo_version_id | The Pexip Infinity software build number. |
| q | The string to lookup in the directory. |
| registration_alias | The alias of the registered client performing the directory lookup. |
| version_id | The Pexip Infinity software version number. |

This example shows a GET request for directory information where the search string is "ali":

```
GET /example.com/policy/v1/registrations?q=ali&registration_
alias=bob@example.com&limit=10&location=london&node_ip=10.44.155.21&pseudo_version_id=36402.0.0&version_
id=16
```

Directory information response

The response to a directory information request takes the format:

```
{
  "status" : "success",
  "result" : [ <directory_data> ],
  "ignore_local" : True|False
}
```

where:

- "result" is a JSON list containing objects representing individual directory entries. Each JSON object in the list contains the following fields:

| Field name | Required | Description |
|-------------|----------|---|
| username | Yes | The username associated with the device or VMR. This is currently unused by Pexip Infinity. |
| alias | Yes | The alias of the device or VMR. This is the alias Pexip Infinity will use as the <alias> in the URI in a subsequent participant avatar request , and the alias that Pexip Infinity clients will dial. |
| description | Yes | A description or name associated with the alias. |

The policy server should sort the list of aliases into the order in which it wants them to be presented.

- "ignore_local" is optional and defaults to False. If set to True it instructs Pexip Infinity to ignore the aliases of any local services or devices i.e. to only use the aliases returned from the policy server as the directory information. When "ignore_local" is False, Pexip Infinity adds the aliases of any local services or devices to the list of aliases returned by the policy server.

Example directory information responses

This is an example of a JSON dictionary of directory information data containing 2 devices:

```
{
  "status" : "success",
  "result" :
  [
    {
      "username" : "alice",
      "alias" : "alice@example.com",
      "description" : "Alice's VMR",
    },
    {
      "username" : "bob",
      "alias" : "bob@example.com",
      "description" : "Bob's VMR",
    }
  ]
}
```

This is an example response that instructs Pexip Infinity to not supply any directory information at all — no aliases are returned by the policy server and it also tells Pexip Infinity to ignore any local aliases ("ignore_local" is True):

```
{
  "status" : "success",
  "result" : [],
  "ignore_local": True
}
```

Registration alias requests

Used to determine whether a device alias is allowed to register to a Conferencing Node.

Request URI: <policy_server_uri>/policy/v1/registrations/<alias> where <alias> is the alias of the device that is making the registration request.

Pexip Infinity includes the following fields in a registration request:

| Parameter | Description |
|-------------------|---|
| location | The name of the Pexip Infinity system location associated with the Conferencing Node making the request/notification. |
| node_ip | The IP address of the Conferencing Node making the request. |
| protocol | For registration requests this is the registration protocol. Values: "webrtc", "sip", or "h323". |
| pseudo_version_id | The Pexip Infinity software build number. |
| version_id | The Pexip Infinity software version number. |

This example shows a GET request for a registration alias of "alice@example.com":

```
GET /example.com/policy/v1/registrations/alice@example.com?pseudo_version_id=36402.0.0&protocol=webrtc&location=london&version_id=16&node_ip=10.44.155.21
```

Registration alias response

The response to a registration alias request takes the format:

```
{
  "status" : "success",
  "action" : "reject|continue",
  "result" : { <alias_data> }
}
```

where:

- "action" is an optional value that, when included, instructs Pexip Infinity to either reject the registration or use its internal database:
 - "reject" instructs Pexip Infinity to reject the registration
 - "continue" (or any other value except "reject") instructs Pexip Infinity to fall back to its default behavior, which is to check if the alias and optionally any associated credentials are configured in its local database of allowed aliases.
- "result" is an optional value that, when included, is a JSON object of key value pairs that contains the credentials to use to authenticate the registration request. The fields that may be included are:

| Field name | Required | Description |
|---|----------|---|
| username | No | The username associated with the device. |
| password | No | The password associated with the device and username. |
| <i>i</i> Note that the password is sent "in the clear" so we recommend using a secure https connection to your policy server. | | |

Pexip Infinity handles the response as follows:

- If the "action" field is specified, all other fields are ignored and Pexip Infinity performs a "reject" or "continue" as specified by the "action".
- If "status" is "success" (and the "action" field is not specified):
 - If the "result" field (containing username / password credentials) is included, then the device alias may register if it supplies matching credentials.
 - If the "result" field is not included, then the device alias may register without any authentication / credential checking.
- If "status" is not "success" or an error code e.g. 404 is returned (and the "action" field is not specified), then Pexip Infinity will fall back to its own default behavior (to check if the alias and optionally any associated credentials are configured in its local database of allowed aliases).

Example registration alias responses

Allowed alias (with authentication): this is an example response containing the required credentials for an allowed alias (the device must then supply the matching credentials to be permitted to register):

```
{
  "status" : "success",
  "result" : {
    "username" : "alice",
    "password" : "password123"
  }
}
```

Allowed alias (without authentication): this is an example response that allows the alias to register without authentication:

```
{
  "status" : "success",
}
```

Denied alias: this is an example response to reject the requested alias:

```
{  
  "status" : "success",  
  "action" : "reject"  
}
```


Summary of request parameters per request type

This table summarizes which fields are included by Pexip Infinity in each request type:

| Parameter | Description | Service config | Media location | Participant avatar | Directory info | Registration alias |
|----------------------|---|----------------|----------------|--------------------|----------------|--------------------|
| bandwidth | The maximum requested bandwidth for the call. | ✓ | ✓ | ✓ | | |
| call_direction | The direction of the call that triggered the request. Values can be: <ul style="list-style-type: none"> "dial_in": calls in to Pexip Infinity "dial_out": calls dialed out from Pexip Infinity "non_dial": when policy is triggered by other requests that are not related to incoming or outgoing call setup, such as when an H.323 LRQ or a SIP SUBSCRIBE or OPTIONS request is received. | ✓ | ✓ | ✓ | | |
| height * | The required height in pixels of the image to be returned. | | | ✓ * | | |
| limit | The maximum number of results to return (currently this is always set to 10). | | | | ✓ | |
| local_alias | In the context of service configuration requests, this is the incoming alias (typically the alias that the endpoint has dialed). This is the primary item of information that the policy server will use to return appropriate service configuration data. For participant requests (media location and participant avatar) this contains the originally-dialed incoming alias for the associated service. | ✓ | ✓ | ✓ | | |
| location * | The name of the Pexip Infinity system location associated with the Conferencing Node making the request/notification. | ✓ | ✓ | ✓ * | ✓ | ✓ |
| ms-subnet † | The sender's subnet address. | ✓ | ✓ | ✓ | | |
| node_ip * | The IP address of the Conferencing Node making the request. | ✓ | ✓ | ✓ * | ✓ | ✓ |
| p_Asserted-Identity† | The authenticated identity of the user sending the SIP message. | ✓ | ✓ | ✓ | | |
| protocol | The call protocol. Values: "api", "webrtc", "sip", "rtmp", "h323" or "mssip". (Note that the protocol is always reported as "api" when an Infinity Connect client dials in to Pexip Infinity.) For registration requests this is the registration protocol. Values: "webrtc", "sip", or "h323". | ✓ | ✓ | ✓ | | ✓ |
| pseudo_version_id * | The Pexip Infinity software build number. | ✓ | ✓ | ✓ * | ✓ | ✓ |

| Parameter | Description | Service config | Media location | Participant avatar | Directory info | Registration alias |
|---------------------|---|----------------|----------------|--------------------|----------------|--------------------|
| q | The string to lookup in the directory. | | | | ✓ | |
| registration_alias | The alias of the registered client performing the directory lookup. | | | | ✓ | |
| remote_address | The IP address of the remote participant. | ✓ | ✓ | ✓ | | |
| remote_alias | The name of the user or the registered alias of the endpoint. The <code>remote_alias</code> may include scheme information, for example <code>sip:alice@example.com</code> . | ✓ | ✓ | ✓ | | |
| remote_display_name | The display name of the remote participant. | ✓ | ✓ | ✓ | | |
| remote_port | The IP port of the remote participant. | ✓ | ✓ | ✓ | | |
| registered | Whether the remote participant is registered or not. Values: "True" or "False". | ✓ | ✓ | ✓* | | |
| role | The role associated with the <code>remote_alias</code> . Values: "chair" (for Host participants), "guest" or "unknown" (a participant who is at the PIN entry screen). | | | ✓ | | |
| service_name | The service name. This will match the <code>name</code> field returned by the policy server from the original service configuration request. | | ✓ | ✓ | | |
| service_tag | The service tag associated with the <code>service_name</code> parameter. | | ✓ | ✓ | | |
| trigger | The trigger for the policy request. Values: "web", "web_avatar_fetch", "invite", "options", "subscribe", "setup", "arq", "lrq" or "unspecified". | ✓ | ✓ | ✓ | | |
| unique_service_name | The unique name used by Pexip Infinity to identify the service: <ul style="list-style-type: none"> For "gateway" services this is the matching rule name followed by a unique identifier (so as to distinguish between separate calls that match the same rule). For all other services, this is the same as the <code>service_name</code> parameter. | | ✓ | ✓ | | |
| vendor | System details about the remote participant, such as manufacturer name and version number for hard endpoints, or browser and operating system details for soft clients. | ✓ | ✓ | ✓ | | |
| version_id * | The Pexip Infinity software version number. | ✓ | ✓ | ✓* | ✓ | ✓ |

| Parameter | Description | Service config | Media location | Participant avatar | Directory info | Registration alias |
|--|---|----------------|----------------|--------------------|----------------|--------------------|
| width * | The required width in pixels of the image to be returned. | | | ✓ * | | |
| * these are the only parameters included in a participant avatar request when Pexip Infinity is retrieving directory information (as opposed to service participant-related information where all parameters are included) | | | | | | |
| † only included if the request was triggered by a SIP message, and the parameter is included as a field in the SIP message header | | | | | | |

Enabling local policy

Local policy allows you to manipulate service configuration and media location data (that has been provided either via the external policy API, or has been retrieved from Pexip Infinity's own database) by running a jinja2 script against that data.

Local policy currently has a more limited scope in what it can control than external policy, but is easier to implement and runs locally on each Conferencing Node.

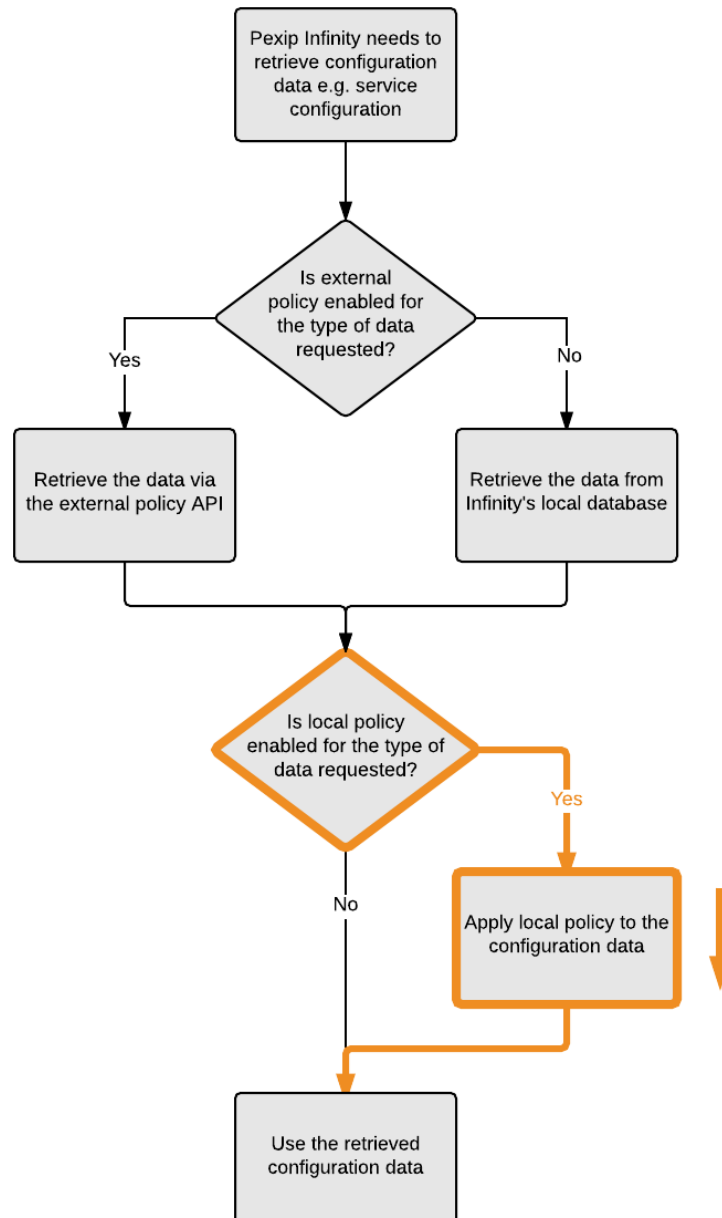
Configuring policy profiles for local policy

You can configure Pexip Infinity to use both external and local policy depending on your requirements. When both external and local policy are enabled, external policy is applied first to retrieve the configuration data from the external system, and then local policy is applied to that retrieved data (which can then conditionally modify that data). The flow chart (right) shows Pexip Infinity's standard processing logic when policy profiles are used and highlights where local policy is applied.

To configure Pexip Infinity to use local policy:

1. Go to **Call Control > Policy Profiles**.
2. Select **Add Policy profile** and then configure that profile:
 - Select **Apply local policy** for the types of configuration data (currently service configuration and media location data types only) that you want to modify via local policy.
 - In the **Script** text box, enter the jinja2 script that you want to execute against that data, and save your changes.
See [Writing local policy scripts](#) for full information about how to construct your jinja2 scripts.
3. Go to **Platform Configuration > Locations**.
4. Select each location in turn and specify the **Policy profile** that the Conferencing Nodes in that location should use when making policy decisions.

For more information on configuring policy profiles and how to combine external policy with local policy, see [Configuring policy profiles](#).



Writing local policy scripts

Local policy allows you to manipulate service configuration and media location data (that has been provided either via the external policy API, or has been retrieved from Pexip Infinity's own database) by running a jinja2 script against that data.

- i** Using local policy requires proficient scripting skills. For large production environments we recommend that you contact your Pexip authorized representative.
- i** We recommend that you test your policy scripts carefully with a variety of different inputs. We will attempt to maintain backwards compatibility in future releases of the Pexip Infinity platform but the introduction of new features may affect the behavior of existing scripts. We strongly recommend that you test your scripts in a lab environment before using them in a production system, particularly after an upgrade to the Pexip Infinity software.

This topic covers:

- [Request types summary](#)
- [Writing a jinja2 script to control call behavior](#)
- [Getting started in constructing a script](#)
- [Supported variables](#)
- [Data manipulation filters](#)
- [Response formats](#) ([Service configuration data responses](#) and [Media location data responses](#))

You can also use the Administrator interface's built-in facility for [Testing local policy scripts](#) and refer to our [Example local policy scripts](#).

Request types summary

The following table shows the types of configuration data that can be modified via local policy, and when that data is requested by Pexip Infinity:

| Type of data | Purpose |
|-----------------------|---|
| Service configuration | Obtains the configuration details of a service. Pexip Infinity typically makes this request when it: <ul style="list-style-type: none">• receives an incoming call request• needs to place a call to a given alias |
| Media location | Obtains the system location to use for media allocation. A Conferencing Node assigned to that location will handle the media for the participant. A media location request is often made after a service configuration data request. |

You may see multiple service configuration and media location requests when handling Infinity Connect participants.

Writing a jinja2 script to control call behavior

Your scripts are configured as part of a policy profile (**Call Control > Policy Profiles**).

Pexip Infinity's local policy scripts are based on a subset of the jinja2 templating language (<http://jinja.pocoo.org/docs/dev/>). .

These scripts are made up of the following elements:

- **variables** that are substituted with values from the source request (`call_info`), or that contain the existing configuration data supplied by external policy or the Pexip Infinity database (`service_config` and `suggested_media_overflow_locations`)
- **filters** that can manipulate text or modify the content of variables or text strings, such as `pex_update` or `pex_to_json`

- **delimiters** such as `{{...}}` and pipes `|` which are used to define filter expressions
- **jinja statements and control structures** (see <http://jinja.pocoo.org/docs/dev/templates/#list-of-control-structures>)

A basic example media location script could be:

```
{
  {% if call_info.location == "Oslo" %}
    "result" : {
      "location" : "Oslo",
      "primary_overflow_location" : "London",
      "secondary_overflow_location" : "New York"
    }
  {% else %}
    "result" : {
      "location" : "New York",
      "primary_overflow_location" : "London",
      "secondary_overflow_location" : "Oslo"
    }
  {% endif %}
}
```

This script tests if the system location handling the incoming call request is "Oslo" and if so sets the location to be used for media to "Oslo", and sets "London" and "New York" as the primary and secondary overflow locations to be used if "Oslo" is out of capacity. For all other system locations handling incoming calls, the location to be used for media will be "New York", with "London" and "Oslo" as the primary and secondary overflow locations.

The following sections provide full information about how to [construct a script](#), use [filters](#), and update/refer to [variables](#).

Getting started in constructing a script

Local policy allows you to manipulate the service configuration and media location data that has already been fetched from Pexip Infinity's own database or has been supplied via the external policy API. And in the case of service configuration data, there may be no data available — for example if a call attempt was made to an alias that does not exist.

This existing configuration data is held in two variables: **service_config** (for service configuration policy) and **suggested_media_overflow_locations** (for media location policy).

Your local policy script will typically use filters to modify some elements of these two variables, or replace the contents of those variables with completely different data, based on certain conditions, or it can choose to pass through the existing data unchanged. Those conditions will typically involve references to that existing data or to the original call information that led to the request to provide service configuration and media location data. That existing call information is held in the **call_info** variable and is explained in [Supported variables](#) below.

The objective of the script is to return the service configuration or media location data to Pexip Infinity. That response to Pexip Infinity i.e. the output/result of the script, must be a JSON object. (Note that the returned data takes the same format as the responses to external policy API requests.)

This is an example response containing service configuration data:

```
{
  "action" : "continue",
  "result" : {
    "service_type" : "conference",
    "name" : "Alice Jones",
    "service_tag" : "abcd1234",
    "pin" : "1357",
  }
}
```

See [Service configuration data responses](#) for full details.

This is an example response containing media location data:

```
{
  "result": {
    "location": "London",
    "primary_overflow_location": "Oslo"
  }
}
```

See [Media location data responses](#) for full details.

i To help you construct and test your scripts, Pexip Infinity has a built-in [script testing facility](#), and there are some [example scripts](#) that illustrate how to structure your scripts.

Supported variables

There is a limited set of system variables ([Configuration variables](#) and [Call information variables](#)) that you can use within your script. Capitalization of variable names is important. The variables **must** be spelled exactly as shown.

Note that these variables are represented in a Python dictionary format (which is very similar, but not identical, to JSON). This is why the `pex_to_json filter` is required if you want to return the content of the configuration variables as the response to a service configuration or media location request.

Configuration variables

The service configuration and media location data that has already been fetched from Pexip Infinity's own database or has been supplied via the external policy API is held in the following variables:

- **service_config**: this holds the existing service configuration data. This variable only applies to scripts run as service configuration policy.
 - When referring to specific service configuration data elements, you must use the **service_config** prefix. For example, use **service_config.pin** to refer to the service PIN, and **service_config.service_type** to refer to the type of service ("conference", "lecture" etc).
 - If no service configuration data has been retrieved — for example if a call attempt was made to an alias that does not exist, this variable will be null, but it can still be updated to contain the required configuration data.
 - See [Service configuration data responses](#) for the full set of service configuration elements within the **service_config** variable. (Note that if you use the `pex_debug_log` filter to log the content of the `service_config` variable, you may see additional fields reported to those listed here. Those additional fields are for information only and may change in future releases.

Example **service_config** data (displayed here in JSON format):

```
"service_config": {
  "allow_guests": true,
  "automatic_participants": [
    {
      "description": "Dial out to VMR owner",
      "local_alias": "meet.alice@example.com",
      "local_display_name": "Alice's VMR",
      "protocol": "sip",
      "remote_alias": "sip:alice@example.com",
      "role": "chair",
      "system_location_name": "London"
    }
  ],
  "description": "Alice Jones personal VMR",
  "enable_overlay_text": true,
  "guest_pin": "5678",
  "name": "Alice Jones",
  "pin": "1234",
  "service_tag": "abcd1234",
}
```

```
"service_type": "conference"
}
```

- **suggested_media_overflow_locations**: this holds the existing media location data. This variable only applies to scripts run as media location policy. When referring to specific media location data elements, you must use the **suggested_media_overflow_locations** prefix followed by the element name — either **location**, **primary_overflow_location** or **secondary_overflow_location** which respectively refer to the principal location and the primary and secondary overflow locations that will handle the call media, for example **suggested_media_overflow_locations.location**.

Example **suggested_media_overflow_locations** data (displayed here in JSON format):

```
"suggested_media_overflow_locations": {
  "location": "New York",
  "primary_overflow_location": "Paris",
  "secondary_overflow_location": "Peckham"
}
```

Call information variables

The following table shows the call information variables that may be available to your script. For each variable the table indicates if it is available to service configuration data requests and/or media location data requests. When using these variables you must use the **"call_info."** prefix. For example to refer to the location associated with the Conferencing Node making the request, use **call_info.location** and to refer to the call protocol use **call_info.protocol**. (Note that the **call_info** variables are the same as those used in external policy API requests.)

| Available call_info variables | Description | Service config | Media location |
|-------------------------------|--|----------------|----------------|
| bandwidth | The maximum requested bandwidth for the call. | ✓ | ✓ |
| call_direction | The direction of the call that triggered the request. Values can be: <ul style="list-style-type: none"> • "dial_in": calls in to Pexip Infinity • "dial_out": calls dialed out from Pexip Infinity • "non_dial": when policy is triggered by other requests that are not related to incoming or outgoing call setup, such as when an H.323 LRQ or a SIP SUBSCRIBE or OPTIONS request is received. | ✓ | ✓ |
| local_alias | In the context of service configuration requests, this is the incoming alias (typically the alias that the endpoint has dialed). This is the primary item of information that the policy script will use to return appropriate service configuration data. For participant requests (media location and participant avatar) this contains the originally-dialed incoming alias for the associated service. | ✓ | ✓ |
| location | The name of the Pexip Infinity system location associated with the Conferencing Node making the request/notification. | ✓ | ✓ |
| ms_subnet † | The sender's subnet address. Note that the value of "ms_subnet" is formatted as a JSON list, for example: ["10.47.5.0"] | ✓ | ✓ |
| node_ip | The IP address of the Conferencing Node making the request. | ✓ | ✓ |
| p_asserted_identity† | The authenticated identity of the user sending the SIP message. Note that the value of "p_asserted_identity" is formatted as a JSON list, for example: ["Alice" <sip:alice@example.com>] | ✓ | ✓ |

| Available call_info variables | Description | Service config | Media location |
|-------------------------------|--|----------------|----------------|
| protocol | The call protocol. Values: "api", "webrtc", "sip", "rtmp", "h323" or "mssip". (Note that the protocol is always reported as "api" when an Infinity Connect client dials in to Pexip Infinity.) | ✓ | ✓ |
| pseudo_version_id | The Pexip Infinity software build number. | ✓ | ✓ |
| remote_address | The IP address of the remote participant. | ✓ | ✓ |
| remote_alias | The name of the user or the registered alias of the endpoint. The remote_alias may include scheme information, for example sip:alice@example.com. | ✓ | ✓ |
| remote_display_name | The display name of the remote participant. | ✓ | ✓ |
| remote_port | The IP port of the remote participant. | ✓ | ✓ |
| registered | Whether the remote participant is registered or not. Values: "True" or "False". | ✓ | ✓ |
| service_name | The service name. This will match the name field returned by the policy script from the original service configuration request. | | ✓ |
| service_tag | The service tag associated with the service_name parameter. | | ✓ |
| trigger | The trigger for the policy request. Values: "web", "web_avatar_fetch", "invite", "options", "subscribe", "setup", "arq", "lrq" or "unspecified". | ✓ | ✓ |
| unique_service_name | The unique name used by Pexip Infinity to identify the service: <ul style="list-style-type: none"> For "gateway" services this is the matching rule name followed by a unique identifier (so as to distinguish between separate calls that match the same rule). For all other services, this is the same as the service_name parameter. | | ✓ |
| vendor | System details about the remote participant, such as manufacturer name and version number for hard endpoints, or browser and operating system details for soft clients. | ✓ | ✓ |
| version_id | The Pexip Infinity software version number. | ✓ | ✓ |

† only included if the request was triggered by a SIP message, and the parameter is included as a field in the SIP message header; in addition the ms_subnet and p_asserted_identity fields use only underscores and lower case characters in their names when used in local policy (normally they are referenced as ms-subnet and p_Asserted-Identity)

For example, the call_info variable could contain the following data (displayed here in JSON format):

```
"call_info": {
  "bandwidth": 0,
  "call_direction": "dial_in",
  "local_alias": "meet.alice.vmr@example.com",
  "location": "New York",
  "node_ip": "10.55.55.101",
  "protocol": "api",
  "pseudo_version_id": "36358.0.0",
  "remote_address": "10.55.55.250",
```

```

"remote_alias": "bob@example.com",
"remote_display_name": "Bob T. User",
"remote_port": 64703,
"trigger": "web",
"vendor": "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/54.0.2840.100
Safari/537.36",
"version_id": "16"
}

```

Data manipulation filters

Pexip Infinity supports a subset of filters from the jinja2 templating language and some custom Pexip filters. These filters are typically used to manipulate data. The most useful filters that you will need when writing local policy scripts are described below.

pex_update

The `pex_update` filter is used to update the `service_config` (service configuration data) and `suggested_media_overflow_locations` (media location data) variables.

Example usage: `{{service_config|pex_update({"pin":"1234", "guest_pin":"","allow_guests": false})}}`

This updates 3 of the data elements in the `service_config` variable. It sets the `pin` to 1234, the `guest_pin` to null and the `allow_guests` flag to false. (See [Service configuration data responses](#) for a list of all of the elements in the `service_config` variable, and [Media location data responses](#) for a list of all of the elements in the `suggested_media_overflow_locations` variable.)

pex_to_json

The `pex_to_json` filter is used to convert the data in the `service_config` and `suggested_media_overflow_locations` variables (which are Python dictionaries) into JSON format (which is the required data structure for the configuration data returned to Using External and Local Policy with Pexip Infinity) when you want to use those variables in a response `result` object.

Example usage: `{{suggested_media_overflow_locations|pex_to_json}}`

The example above converts the `suggested_media_overflow_locations` variable into JSON format.

Example usage: `{{service_config|pex_update({"participant_limit":10, "ivr_theme_name":"funky"})|pex_to_json}}`

This second example combines the `pex_update` filter with the `pex_to_json` filter. It makes updates to the `service_config` variable and then converts it to JSON.

Note that you do not need to use the `pex_to_json` filter if you are constructing the response `result` object directly in JSON format (e.g. as shown in the [Example service configuration data response](#)).

pex_in_subnet

The `pex_in_subnet` filter is used to test whether a given address is within one or more subnets. This filter could be useful if, for example, you want to place media in a particular location based upon the network location of the participant.

It takes as input the address you want to test, and one or more subnet ranges, and returns either True or False. For example:

- `{{pex_in_subnet("10.47.0.1", ["10.47.0.0/16", "10.147.0.0/16"])}}` returns True
- `{{pex_in_subnet("10.44.0.1", ["10.47.0.0/16", "10.147.0.0/16"])}}` returns False
- `{{pex_in_subnet("2001:658:22a:cafe:ffff:ffff:ffff:ffff", ["2001::/16", "2002::/16"])}}` returns True
- `{{pex_in_subnet("2000:758:23a:beef:ffff:ffff:ffff:ffff", ["2001::/16", "2002::/16"])}}` returns False

For practical usage you will most likely want to refer to the calling participant's address (`call_info.remote_address`) instead of literally specifying the address to be tested, for example: `{{pex_in_subnet(call_info.remote_address, ["10.44.0.0/16"])}}`


See [Test whether a participant's address is within a particular subnet](#) for a more advanced example that shows how you can make multiple tests against multiple locations with multiple subnets.

pex_debug_log

The `pex_debug_log` filter can be used to help debug your script. It writes debug messages to the Pexip Infinity support log. You can include literal text and variables.

Example usage: `{{pex_debug_log("Media location policy ", call_info.location, " protocol", call_info.protocol) }}`

This will generate a support log entry in the style of: Name="support.jinja2" pex_debug_log Detail="Media location policy ,Oslo, protocol,sip"

 To avoid filling the support log and causing it to rotate, remove all `pex_debug_log` filters from your scripts as soon as they are working correctly.

Response formats

The response to Pexip Infinity i.e. the output/result of the script, must be a JSON object.

The following sections explain how that response must be structured for service configuration data and for media location data. Note that the fields in a JSON object can be supplied in any order, and that the returned data takes the same format as the responses to external policy API requests.

Service configuration data responses

The response to a service configuration request takes the basic format:

```
{
  "action" : "reject|continue",
  "result" : { <service_configuration_data> }
}
```

where:

- "action" instructs Pexip Infinity how to proceed
 - "reject" instructs Pexip Infinity to reject the call immediately (i.e. return "conference not found")
 - "continue" use the data supplied in the "result" field
- "result" is a JSON object of multiple key value pairs that describes the service. Some data fields are required, and some are optional. The fields expected by Pexip Infinity in the response depend upon the returned `service_type` — either "conference" or "lecture" for a Virtual Meeting Room or Virtual Auditorium, "gateway" for a Pexip Distributed Gateway call, "two_stage_dialing" for a Virtual Reception, or "test_call" for a Test Call Service.

The fields expected in the response for each service type are described below:

Virtual Meeting Room / Virtual Auditorium service types response fields

Pexip Infinity expects the following fields to be returned for a `service_type` of "conference" (VMR) or "lecture" (Virtual Auditorium):

| Field name | Required | Type | Description |
|--------------|----------|--------|---|
| service_type | Yes | String | The type of service, in this case: <ul style="list-style-type: none">• "conference": a Virtual Meeting Room, or• "lecture": a Virtual Auditorium |

| Field name | Required | Type | Description |
|--|----------|---------|--|
| name | Yes | String | The name of the service. You could, for example, use the <code>local_alias</code> received in the configuration request. Pexip Infinity will subsequently use this name — as the <code>service_name</code> parameter — in subsequent requests. |
| description | No | String | A description of the service. |
| service_tag | Yes | String | A unique identifier used to track usage of this service. |
| pin | No | String | Host PIN — the secure access code for Host participants. |
| allow_guests | No | Boolean | Whether to distinguish between Host and Guest participants: <ul style="list-style-type: none"> <code>true</code>: the conference has two types of participants: Hosts and Guests. The <code>pin</code> to be used by Hosts must be specified. A <code>guest_pin</code> can optionally be specified; if a <code>guest_pin</code> is not specified, Guests can join without a PIN. <code>false</code>: all participants have Host privileges Default: <code>false</code> |
| guest_pin | No | String | Guest PIN — the secure access code for Guest participants. |
| guests_can_present | No | Boolean | Controls whether the Guests in the conference are allowed to present content. <ul style="list-style-type: none"> <code>true</code>: Guests and Hosts can present into the conference <code>false</code>: only Hosts can present Default: <code>true</code> |
| max_callrate_in | No | Integer | The maximum media bandwidth in bps that Pexip Infinity will receive from each individual participant dialed in to the service. Range 128000 to 4096000 bps. |
| max_callrate_out | No | Integer | The maximum media bandwidth in bps that Pexip Infinity will send to each individual participant dialed in to the service. Range 128000 to 4096000 bps. |
| participant_limit | No | Integer | The maximum number of participants allowed to join the service. |
| ivr_theme_name | No | String | The name of the theme to use with the service. If no theme is specified, the default Pexip theme is used. |
| automatic_participants | No | List | A list of participants to dial automatically when the conference starts. Each participant in the list contains a number of fields as described in Automatically dialed participants (ADP) fields . |
| view (applies to service_type of "conference" only) | No | String | The layout seen by all participants: <ul style="list-style-type: none"> <code>"one_main_zero_pips"</code>: full-screen main speaker only <code>"one_main_seven_pips"</code>: large main speaker and up to 7 other participants <code>"one_main_twentyone_pips"</code>: main speaker and up to 21 other participants <code>"two_mains_twentyone_pips"</code>: two main speakers and up to 21 other participants Default: <code>"one_main_seven_pips"</code> |

| Field name | Required | Type | Description |
|---|----------|---------|---|
| host_view (applies to service_type of "lecture" only) | No | String | <p>The layout seen by Hosts:</p> <ul style="list-style-type: none"> "one_main_zero_pips": full-screen main speaker only "one_main_seven_pips": large main speaker and up to 7 other participants "one_main_twentyone_pips": main speaker and up to 21 other participants "two_mains_twentyone_pips": two main speakers and up to 21 other participants <p>Default: "one_main_seven_pips"</p> |
| guest_view (applies to service_type of "lecture" only) | No | String | <p>The layout seen by Guests:</p> <ul style="list-style-type: none"> "one_main_zero_pips": full-screen main speaker only "one_main_seven_pips": large main speaker and up to 7 other participants "one_main_twentyone_pips": main speaker and up to 21 other participants "two_mains_twentyone_pips": two main speakers and up to 21 other participants <p>Default: "one_main_seven_pips"</p> <p>Only Hosts are displayed. Guests can hear but not see any of the other Guests.</p> |
| enable_overlay_text | No | Boolean | <p>If participant name overlays are enabled, the display name or alias of the participants (the current main speaker and thumbnails) is shown in a text overlay along the bottom of their video image.</p> <ul style="list-style-type: none"> true: participant names are shown false: participant names are not shown <p>Default: false</p> |
| locked | No | Boolean | <p>Whether to lock the conference on creation:</p> <ul style="list-style-type: none"> true: the conference will be locked on creation false: the conference will not be locked <p>Note that this field has no effect on the conference if it is already running.</p> <p>Default: false</p> |
| call_type | No | String | <p>The call capability of the conference. It can be limited to:</p> <ul style="list-style-type: none"> "video": main video plus presentation "video-only": main video only "audio": audio-only <p>Default: "video"</p> |
| enable_chat | No | String | <p>Whether chat messaging is enabled for the conference:</p> <ul style="list-style-type: none"> "default": as per the global configuration setting "yes": chat is enabled "no": chat is disabled <p>Default: "default"</p> |
| primary_owner_email_address | No | String | The email address of the owner of the VMR. |

| Field name | Required | Type | Description |
|--|----------|---------|--|
| force_presenter_into_main (applies to service_type of "lecture" only) | No | Boolean | Controls whether the Host who is presenting is locked into the main video position: <ul style="list-style-type: none"> true: the Host sending the presentation stream will always hold the main video position false: the main video position is voice-switched Default: false |
| mute_all_guests (applies to service_type of "lecture" only) | No | Boolean | Controls whether to mute guests when they first join the conference: <ul style="list-style-type: none"> true: mute Guests when they first join the conference false: do not mute Guests when they first join the conference Default: false |

Pexip Distributed Gateway service type response fields

Pexip Infinity expects the following fields to be returned for a **service_type** of "gateway":

| Field name | Required | Type | Description |
|--------------------|----------|---------|--|
| service_type | Yes | String | The type of service, in this case: <ul style="list-style-type: none"> "gateway": a Pexip Distributed Gateway call |
| name | Yes | String | The name of the service. Ensure that all gateway service instances have a unique name to avoid conflicts between calls. Pexip Infinity will subsequently use this name — as the service_name parameter — in subsequent requests. |
| description | No | String | A description of the service. |
| service_tag | Yes | String | A unique identifier used to track usage of this service. |
| max_callrate_in | No | Integer | The maximum media bandwidth in bps that Pexip Infinity will receive from each individual participant dialed in to the service. Range 128000 to 4096000 bps. |
| max_callrate_out | No | Integer | The maximum media bandwidth in bps that Pexip Infinity will send to each individual participant dialed in to the service. Range 128000 to 4096000 bps. |
| ivr_theme_name | No | String | The name of the theme to use with the service. If no theme is specified, the default Pexip theme is used. |
| remote_alias | Yes | String | The alias of the endpoint to call. The alias can include scheme information, for example sip:alice@example.com , but the call will always be made using the specified outgoing_protocol . |
| local_alias | Yes | String | The calling or "from" alias. This is the alias that the recipient would use to return the call. |
| local_display_name | No | String | The display name of the calling alias. |

| Field name | Required | Type | Description |
|------------------------|----------|--------|--|
| outgoing_protocol | Yes | String | The protocol to use to place the outgoing call: <ul style="list-style-type: none"> "h323": an H.323 call; you can also optionally nominate an <code>h323_gatekeeper_name</code> "sip": a SIP call; you can also optionally nominate a <code>sip_proxy_name</code> "mssip": a Microsoft Skype for Business / Lync call; you can also optionally nominate an <code>mssip_proxy_name</code> and a <code>turn_server_name</code> "rtmp": uses the RTMP protocol; typically this is used for content streaming |
| outgoing_location_name | No | String | The name* of the location of a Conferencing Node from which to place the call. |
| h323_gatekeeper_name | No | String | The name* of the H.323 gatekeeper to use to place the outgoing call. DNS is used if no gatekeeper is specified. |
| sip_proxy_name | No | String | The name* of the SIP proxy to use to place the outgoing call. DNS is used if no proxy is specified. |
| mssip_proxy_name | No | String | The name of the Skype for Business / Lync server to use to place the outgoing call. DNS is used if no server is specified. |
| turn_server_name | No | String | The name* of the TURN server to offer to external SIP and WebRTC endpoints that are ICE-enabled (such as Skype for Business / Lync clients and Infinity Connect WebRTC clients). |
| stun_server_name | No | String | The name* of the STUN server to use when placing calls to external SIP and WebRTC endpoints that are ICE-enabled (such as Skype for Business / Lync clients and Infinity Connect WebRTC clients). |
| call_type | No | String | The call capability of the outbound call: <ul style="list-style-type: none"> "video": main video plus presentation "video-only": main video only "audio": audio-only "auto": match the capability of the outbound call to that of the inbound call Default: "video" |
| called_device_type | No | String | The device or system to which to route the call: <ul style="list-style-type: none"> "external": route the call to a matching registered device if it is currently registered, otherwise attempt to route the call via an external system "registration": route the call to a matching registered device only (providing it is currently registered) "mssip_conference_id": route the call via a Skype for Business / Lync server to a Sfb/Lync meeting where the <code>remote_alias</code> is a Sfb/Lync meeting Conference ID "mssip_server": route the call via a Skype for Business / Lync server to a Sfb/Lync client or meeting Default: "external" |

* The returned "name" must match the name of the location, H.323 gatekeeper, SIP proxy etc. configured within Pexip Infinity.

Virtual Reception service type response fields

Pexip Infinity expects the following fields to be returned for a **service_type** of "two_stage_dialing" (Virtual Reception):

| Field name | Required | Type | Description |
|----------------------|----------|---------|--|
| service_type | Yes | String | The type of service, in this case: <ul style="list-style-type: none"> "two_stage_dialing": a Virtual Reception |
| name | Yes | String | The name of the service. You could, for example, use the local_alias received in the configuration request. Pexip Infinity will subsequently use this name — as the service_name parameter — in subsequent requests. |
| description | No | String | A description of the service. |
| service_tag | Yes | String | A unique identifier used to track usage of this service. |
| max_callrate_in | No | Integer | The maximum media bandwidth in bps that Pexip Infinity will receive from each individual participant dialed in to the service. Range 128000 to 4096000 bps. |
| max_callrate_out | No | Integer | The maximum media bandwidth in bps that Pexip Infinity will send to each individual participant dialed in to the service. Range 128000 to 4096000 bps. |
| ivr_theme_name | No | String | The name of the theme to use with the service. If no theme is specified, the default Pexip theme is used. |
| call_type | No | String | The call capability of the reception. It can be limited to: <ul style="list-style-type: none"> "video": main video plus presentation "video-only": main video only "audio": audio-only Default: "video" |
| mssip_proxy_name | No | String | The name of the Skype for Business / Lync server to use to place the outgoing call. DNS is used if no server is specified. |
| system_location_name | No | String | This is an optional field used in conjunction with the mssip_proxy_name setting. If specified, a Conferencing Node in this system location will perform the SfB/Lync Conference ID lookup on the SfB/Lync server. If a location is not specified, the IVR ingress node will perform the lookup. |
| match_string | No | String | An optional regular expression used to match against the alias entered by the caller into the Virtual Reception. If the entered alias does not match the expression, the Virtual Reception will not route the call. |
| replace_string | No | String | An optional regular expression used to transform the alias entered by the caller into the Virtual Reception. (Only applies if a regex match string is also configured and the entered alias matches that regex.) Leave this field blank if you do not want to change the alias entered by the caller. |

Test Call Service type response fields

Pexip Infinity expects the following fields to be returned for a **service_type** of "test_call" (Test Call Service):

| Field name | Required | Type | Description |
|-----------------|----------|---------|---|
| service_type | Yes | String | The type of service, in this case: <ul style="list-style-type: none"> "test_call": a Test Call Service |
| name | Yes | String | The name of the service. You could, for example, use the local_alias received in the configuration request. Pexip Infinity will subsequently use this name — as the service_name parameter — in subsequent requests. |
| description | No | String | A description of the service. |
| service_tag | Yes | String | A unique identifier used to track usage of this service. |
| max_callrate_in | No | Integer | The maximum media bandwidth in bps that Pexip Infinity will receive from each individual participant dialed in to the service. Range 128000 to 4096000 bps. |
| ivr_theme_name | No | String | The name of the theme to use with the service. If no theme is specified, the default Pexip theme is used. |
| call_type | No | String | The call capability of the service. It can be limited to: <ul style="list-style-type: none"> "video": main video plus presentation "video-only": main video only "audio": audio-only Default: "video" |

Automatically dialed participants (ADP) fields

The **automatic_participants** list field, which may be included in the service configuration response for a **service_type** of "conference" (VMR) or "lecture" (Virtual Auditorium), contains the following fields per participant:

| Field name | Required | Type | Description |
|---------------------|----------|--------|---|
| remote_alias | Yes | String | The alias of the endpoint to call. |
| remote_display_name | No | String | An optional friendly name for this participant. This may be used instead of the participant's alias in participant lists and as a text overlay in some layout configurations. |
| local_alias | Yes | String | The calling or "from" alias. This is the alias that the recipient would use to return the call. |
| local_display_name | No | String | The display name of the calling or "from" alias. |
| description | No | String | An optional description of the Automatically Dialed Participant. Note that the description is for information purposes only. It has no subsequent effect in call processing. |

| Field name | Required | Type | Description |
|-----------------------|----------|---------|---|
| routing | No | String | <p>Specifies how to route the call:</p> <ul style="list-style-type: none"> "manual": uses the requested protocol and the defaults for the specified <code>system_location_name</code>. "routing_rule": routes the call according to the configured Call Routing Rules— this means that the dialed alias must match an outgoing Call Routing Rule for the call to be placed (using the protocols and call control systems etc. as configured for that rule). <p>Default: "manual"</p> |
| protocol | Yes | String | <p>The protocol to use to place the outgoing call:</p> <ul style="list-style-type: none"> "h323": an H.323 call "sip": a SIP call "mssip": a Microsoft Skype for Business / Lync call "rtmp": uses the RTMP protocol; typically this is used for content streaming <p>Calls are routed to externally-located participants based on the configuration of the system location used to place the call.</p> |
| role | Yes | String | <p>The level of privileges the participant has in the conference:</p> <ul style="list-style-type: none"> "chair": the participant has Host privileges "guest": the participant has Guest privileges |
| dtmf_sequence | No | String | An optional DTMF sequence to be transmitted after the call to the dialed participant starts. |
| system_location_name | No | String | The location of the Conferencing Node from which to place the call. |
| streaming | No | Boolean | <p>Identifies the dialed participant as a streaming or recording device.</p> <ul style="list-style-type: none"> true: the participant is a streaming or recording device false: the participant is not a streaming or recording device <p>Default: false</p> |
| keep_conference_alive | No | String | <p>Determines whether the conference continues when all other non-ADP participants have disconnected:</p> <ul style="list-style-type: none"> "keep_conference_alive": the conference continues to run until this participant disconnects (applies to Hosts only). "keep_conference_alive_if_multiple": the conference continues to run as long as there are two or more "keep_conference_alive_if_multiple" participants and at least one of them is a Host. "keep_conference_alive_never": the conference terminates automatically if this is the only remaining participant. <p>Default: "keep_conference_alive_if_multiple"</p> <p>For more information, see https://docs.pexip.com/admin/automatically_terminate.htm.</p> |

| Field name | Required | Type | Description |
|------------------|----------|--------|--|
| call_type | No | String | The call capability of the participant. It can be limited to: <ul style="list-style-type: none"> "video": main video plus presentation "video-only": main video only "audio": audio-only Default: "video" |
| presentation_url | No | String | This additional parameter can be specified for RTMP calls to send the presentation stream to a separate RTMP destination. |

Example service configuration data responses

Response containing service configuration data

This is an example response of service configuration data for a "conference" service type:

```
{
  "action" : "continue",
  "result" : {
    "service_type" : "conference",
    "name" : "Alice Jones",
    "service_tag" : "abcd1234",
    "description" : "Alice Jones personal VMR",
    "pin" : "1234",
    "allow_guests" : true,
    "guest_pin" : "5678",
    "view" : "one_main_zero_pips",
    "enable_overlay_text" : true,
    "automatic_participants" :
    [
      {
        "remote_alias" : "sip:alice@example.com",
        "remote_display_name" : "Alice",
        "local_alias" : "meet.alice@example.com",
        "local_display_name" : "Alice's VMR",
        "protocol" : "sip",
        "role" : "chair",
        "system_location_name" : "London"
      },
      {
        "remote_alias" : "rtmp://example.com/live/alice_vmr",
        "local_alias" : "meet.alice@example.com",
        "local_display_name" : "Alice's VMR",
        "protocol" : "rtmp",
        "role" : "guest",
        "streaming" : true
      }
    ]
  }
}
```

Response instructing Pexip Infinity to reject the call

This is an example response that tells Pexip Infinity to reject a call.

```
{
  "action" : "reject"
}
```

Media location data responses

The response to a media location request takes the format:

```
{
  "result" : { <location_data> }
}
```

where "result" is a JSON object of key value pairs that describes the media location to use. The fields that may be included are:

| Field name | Required | Description |
|-----------------------------|----------|---|
| location | Yes | The name* of the principal location to use for media allocation. A Conferencing Node assigned to that location will handle the media for the participant. |
| primary_overflow_location | No | The name* of the system location to handle the media if the principal location has reached its capacity. |
| secondary_overflow_location | No | The name* of the system location to handle the media if both the principal location and the primary overflow location have reached their capacity. |

* The returned "name" must match the name of a location configured within Pexip Infinity. If any of the location names included in the response do not match a configured location within Pexip Infinity, the entire response is deemed to have failed and Pexip Infinity will fall back to its own default behavior for that request.

Example media location data response

This is an example response containing media location data:

```
{
  "result" : {
    "location" : "London",
    "primary_overflow_location" : "Oslo"
  }
}
```

Dialing out from conference

Pexip Infinity makes a service configuration request if it dials out from a conference in order to invite a participant to join (where the `call_direction` parameter will be `dial_out`). In these cases, the response to the service configuration request must match the existing service data (i.e. the same `name`, `service_type` and so on).

When dialing out, the only configuration you can control via policy is the media location — you can do this in the response to the media location request that follows the service configuration request.


Testing local policy scripts

To help you construct and test your scripts, Pexip Infinity has a built-in script testing facility. It allows you to provide some test configuration data, run your script against that test data and check the output produced by your script.

To test your scripts:

1. Go to **Call Control > Policy Profiles** and select the policy profile containing the script you want to test.
(Note that you must select an existing profile.)
2. Select **Test local service configuration policy** or **Test local media location policy** as appropriate (at the bottom of the page).
3. You are taken to a script testing page where you can provide test input data and edit your script:
 - The **Input** field contains some example **call_info** and either some **service_config** data (for service configuration scripts) or some **suggested_media_overflow_locations** data (for media location scripts), in JSON format, that you can use as input data to test your script. You can use this data as is, or change the data to suit your needs, by adding or removing data elements. Ensure that you use valid **call_info** and **service_config** / **suggested_media_overflow_locations** data elements and maintain the correct JSON formatting.
Note that you cannot include **service_config** data as input to a media location script, and you cannot use **suggested_media_overflow_locations** data as input to a service configuration script.
 - The **Script** field is initially populated with your current script from your policy profile. You can edit the script as required.
 - The **Result** field shows the result of running the input data against your script. If there is an error when running the script this field will contain a "reject" action.
 - The **Diagnostics** field contains any additional information (if available) pertaining to errors arising from running the test script against the test input.
4. To test your script, edit the **Input** and **Script** fields as required and then select **Test local service configuration policy** / **Test local media location policy** (at the bottom of the page).
5. Check the **Result** and **Diagnostic** fields to see the effect of your script.
6. You can continue to edit the **Input** and **Script** fields and test your script until you are happy that the script is performing as expected.
7. When you have finished testing your script:
 - Select **Save changes and return** to save the current contents of your script to your policy profile.
 - Select **Cancel** to return to your policy profile without saving the changes to your script.

Checking call information (call_info)

-  The contents of the **call_info** variable can vary within your Pexip Infinity system depending on the types of devices and call protocols in use. A good way to identify what information is actually being presented to your script is to include a **pex_debug_log** filter such as:

```
{{pex_debug_log("Call information ", call_info) }}
```

You can then make some test calls and look in the support log to see the actual call information that is being passed through to your script. To see the call information data go to **Status > Support Log** and then search for entries that contain **pex_debug_log**. See [Basic pass-through service configuration script with call_info debug line](#) for a full example script.

Example local policy scripts

Here are some example local policy scripts that illustrate how to structure a script, manipulate variables and format the data responses.

You can use and adapt these scripts as appropriate for your own environment.

Minimum basic pass-through service configuration script

This is the smallest "no changes" service configuration script that you can use. It will allow a call to continue if it was going to be allowed anyway — and reject it if it was going to be rejected.

We recommend that you use this as the basis for your own scripts and add extra logic to get your desired functionality.

```
{
  {% if service_config %}
    "action" : "continue",
    "result" : {{service_config|pex_to_json}}
  {% else %}
    "action" : "reject",
    "result" : {}
  {% endif %}
}
```

Basic pass-through service configuration script with call_info debug line

This script extends the minimum "no impact" service configuration script above by adding a `{{pex_debug_log("Call information ", call_info) }}` line.

When this script runs it will not change any configuration data but it will record the contents of the `call_info` variable — all of the information relating to the call being processed — to the support log. This allows you to analyze the nature of the call information for different call types and help you construct your script appropriately. To see the call information data go to **Status > Support Log** and then search for entries that contain `pex_debug_log`.

i To avoid filling the support log and causing it to rotate, remove all `pex_debug_log` filters from your scripts as soon as they are working correctly.

```
{
  {{pex_debug_log("Call information ", call_info) }}
  {% if service_config %}
    "action" : "continue",
    "result" : {{service_config|pex_to_json}}
  {% else %}
    "action" : "reject",
    "result" : {}
  {% endif %}
}
```

Minimum basic pass-through media location script

This is the smallest "no changes" media location script that you can use. It simply returns the original set of suggested_media_overflow_locations without modification.

```
{
  "result" : {{suggested_media_overflow_locations|pex_to_json}}
}
```

Unconditionally nominate media locations

This media location script sets the location to be used for media to "Oslo", and sets "London" and "New York" as the primary and secondary overflow locations if "Oslo" is out of capacity. These locations will be used for every call / media location request, regardless of where call signaling is received.

```
{
  "result" : {
    "location" : "Oslo",
    "primary_overflow_location" : "London",
    "secondary_overflow_location" : "New York"
  }
}
```

Nominate a media location for RTMP streaming

This media location script explicitly sets the location to use for media to "DMZ" if the call being made is an outbound RTMP call (and also overrides any original primary and secondary overflow locations). All other call types will continue to use the original media locations.

```
{
  {% if
    call_info.protocol == "rtmp" and
    call_info.call_direction == "dial_out"
  %}
    "result" : {
      "location" : "DMZ",
      "primary_overflow_location" : "Location B",
      "secondary_overflow_location" : "Location C"
    }
  {% else %}
    "result" : {{suggested_media_overflow_locations|pex_to_json}}
  {% endif %}
}
```

Remove PIN based on location

This service configuration script removes the PIN for participants in "Location A" or "Location C" (these might, for example, be "internal" locations) but keeps any existing PIN requirements for participants in other locations.

```
{
  {% if service_config %}
    "action" : "continue",
    {% if call_info.location == "Location A" or call_info.location == "Location C" %}
      "result" : {{service_config|pex_update({"pin":"","guest_pin":"","allow_guests" : False})|pex_to_
json}}
    {% else %}
      "result" : {{service_config|pex_to_json}}
    {% endif %}
  {% else %}
    "action" : "reject",
    "result" : {}
  {% endif %}
}
```

Remove PIN if device is registered

This service configuration script removes the PIN for participants if their device is registered to a Conferencing Node, but keeps any existing PIN requirements for participants in other locations.

```
{
  {% if service_config %}
    "action" : "continue",
    {% if call_info.registered %}
      "result" : {{service_config|pex_update({"pin":"","guest_pin":"","allow_guests" : False})|pex_to_
json}}
    {% else %}
      "result" : {{service_config|pex_to_json}}
    {% endif %}
  {% else %}
    "action" : "reject",
    "result" : {}
  {% endif %}
}
```


Inject an ADP into every conference (with a debug line for service_config)

This service configuration script adds an Automatically Dialed Participant (ADP) into every conference. Note that this script will override any existing ADPs that may be present in the `service_config` variable.

This example script also demonstrates how to use the `pex_debug_log` filter. It writes the original contents of the `service_config` variable to the support log.

- i** To avoid filling the support log and causing it to rotate, remove all `pex_debug_log` filters from your scripts as soon as they are working correctly.

```
{
  {% if service_config %}
    {{pex_debug_log("service_config=", service_config) }}
    "action" : "continue",
    "result" : {{service_config|pex_update({"automatic_participants" : [{"remote_
alias":"participant@domain.com","local_alias":"policyuser@domain.com","local_display_name":"Local policy
ADP","description":"Dial out to an ADP","protocol":"sip","role":"chair","system_location_name":"Location
A" }]} )|pex_to_json}}
  {% else %}
    "action" : "reject",
    "result" : {}
  {% endif %}
}
```

Reject specified User Agents

This service configuration script rejects calls from (potentially) malicious User Agents such as "sipvicious" (User Agent "friendly-scanner"). Any calls from User Agents that are in the `suspect_uas` list will be rejected. You can edit the `suspect_uas` as required for your environment if appropriate.

```
{
  {% NOTE: not all of the UAs listed are always malicious - they have legitimate uses so you may wish to
adapt the list to your particular environment/usage %}
  {% NOTE: "cisco" is not used by genuine Cisco equipment - rather by an openH323 based VOIP scanner
trying to pass itself off as something respectable %}
  {% set suspect_uas = [ "cisco", "friendly-scanner", "sipcli", "sipvicious", "sip-scan", "sipsak",
"sundayddr", "iWar", "CSipSimple", "SIVuS", "Gulp", "sipv", "smap", "friendly-request",
"VaxIPUserAgent", "VaxSIPUserAgent", "siparmyknife", "Test Agent" ] %}

  {% if service_config %}
    {% if call_info.vendor in suspect_uas %}
      "action" : "reject",
      "result" : {}
    {% else %}
      "action" : "continue",
      "result" : {{service_config|pex_to_json}}
    {% endif %}
  {% else %}
    "action" : "reject",
    "result" : {}
  {% endif %}
}
```

Test whether a participant's address is within a particular subnet

You can use the `pex_in_subnet` filter to test whether a given address is within one or more subnets. This filter could be useful if, for example, you want to place media in a particular location based upon the network location of the participant.

This example template structure sets an address map variable (`nl_map`) to contain a range of subnets associated with multiple locations, and then uses that variable in multiple `pex_in_subnet` tests to see if the calling participant's address (`call_info.remote_address`) is within one of those subnets:

```
{% set nl_map = {  "Oslo" : [ "10.47.0.0/16", "10.147.0.0/16", "10.247.200.0/24" ],
                  "New York" : [ "10.1.0.0/16", "10.201.5.0/23"],
                  "Sydney" : [ "10.61.0.0/16" ],
                  "London" : [ "10.44.0.0/16" ] } %}
{% if pex_in_subnet(call_info.remote_address, nl_map["Oslo"]) %}
  {# Apply Norwegian rules #}
{% elif pex_in_subnet(call_info.remote_address, nl_map["New York"]) %}
  {# Apply American rules #}
{% else %}
  {# Do the default thing #}
{% endif %}
```

Lock a conference when the first participant connects

This example shows a way of locking a conference when the first participant (regardless of Host/Guest) connects. When a conference is locked a Host can manually unlock the conference with *7 or use Infinity Connect to let someone in.

It works by looking at the **Service tag** property of the VMR and if the tag value starts with "locked" it will lock the conference. If it doesn't start with "locked" it will not lock the conference. This requires that you set in advance the **Service tag** to "locked" of the VMRs that you want to automatically lock when the first participant joins. (You can use a different value to "locked" in the service tag if required, or test with "endswith" if you want to use the last word in the service tag.)

```
{
  {% if service_config %}
    "action" : "continue",
    {% if service_config.service_type == "conference" and service_config.service_tag.startswith
("locked") %}
      "result" : {{service_config|pex_update({"locked":true })|pex_to_json}}
    {% else %}
      "result" : {{service_config|pex_to_json}}
    {% endif %}
  {% else %}
    "action" : "reject",
    "result" : {}
  {% endif %}
}
```