

IP Network Enabler

This chapter describes the StarOS IP Network Enabler (IPNE) feature. It describes how the feature works, and how to configure and monitor IPNE.

- Feature Description, on page 1
- How it Works, on page 2
- Configuring the IPNE Feature, on page 8
- Monitoring the IPNE Service, on page 9

Feature Description

This section provides a description of the IPNE feature.

IPNE (IP Network Enabler) is a MINE client component running on various network nodes within operator's network (P-GW, GGSN, HA, or HNBGW), to collect and distribute session/network information to MINE servers. The MINE cloud service provides a central portal for wireless operators and partners to share and exchange session and network information to realize intelligent services.

The information is shared between the MINE server and IPNE service in the form of XML data. The core object in the IPNE service is the XMPP protocol engine. There is one XMPP protocol engine instance for each configured MINE server peer. The engine implements the XMPP protocol using FSM.

All information that is shared is derived from the context at that instance in time. An IPNE service level scheduler is also implemented to rate-control the feed and notification activities on all the handles to avoid overload which would affect call processing and data path performance.

Relationships to Other Features

This section describes how the IPNE service is related to other features.

One of the following GW services must be configured on the StarOS before IPNE can be configured:

- GGSN
- HA
- HNBGW
- P-GW

Refer to the *GGSN Administration Guide*, the *HA Administration Guide*, the *HNBGW Administration Guide* and the *P-GW Administration Guide* for configuration procedures.

The MINE cloud service provides a central portal for wireless operators and partners to share and exchange session/network information to realize intelligent services. A MINE client component is running on various network nodes within operator's network, e.g. PGW, HA, to collect and distribute session/network information to MINE servers. The client is IPNE.

The IPNE client runs on the StarOS as a configurable service. The Enhanced Charging Service (ECS) component interacts with the IPNE client in order to fulfill the defined requirements.

For best IPNE performance, the ECS component should provide the following functionality:

- Flow information parameters should be provided by ECS to IPNE:
  - Tuple information
  - URL
  - User Agent
  - Application protocol
  - Flow creation time

NBR information parameters should be provided by ECS to IPNE:

- NAT-IP address
- Start Port
- End Port

ECS should provide the above parameters for all active flows in a response corresponding to the query from the MINE server indexed on the subscriber's call id.

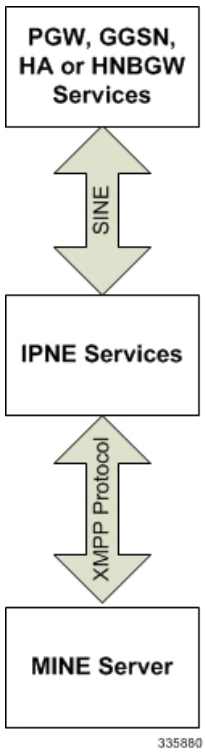
For the subscription that is installed by the IPNE client on a subscriber's call id, ECS should send a notification message to the IPNE client whenever a subscribed trigger is detected.

How it Works

IPNE

The following diagram describes the architecture for the IPNE interface. The session manager and IPNE will interact via the SINE interface. The information will be exchanged between the modules in the form of clp handles. For each session one IPNE handle is created. The information is stored in a local database on the IPNE client side.

Figure 1: High-Level IPNE Architecture



The interaction takes place at the time of:

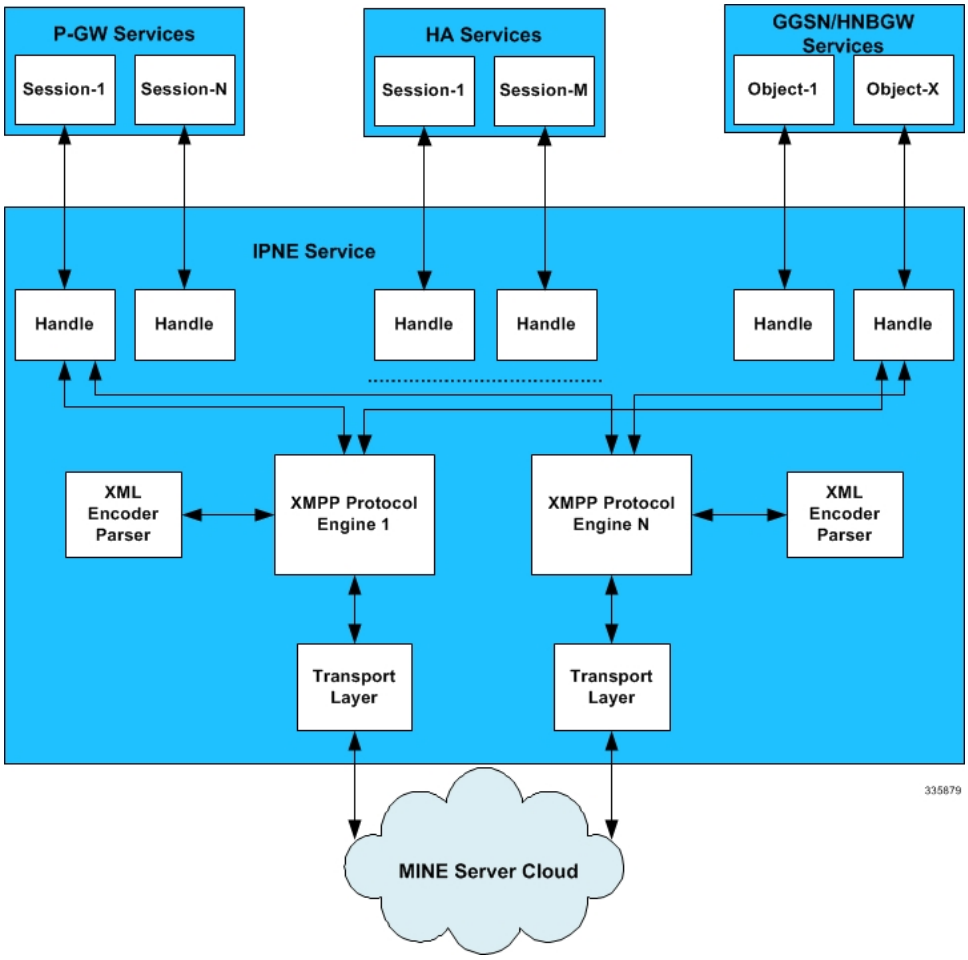
- Session setup so as to add the session information at the IPNE side
- To pass the feed messages to the MINE server
- While responding to the query request sent by the MINE server.
- Subscription notification from IPNE client to MINE server

The MINE server and IPNE client interact with each other for all procedures using the XMPP protocol over the SINE interface. The information stored at the IPNE client side is converted to XML format and then passed on to the MINE server. Upon receiving the messages (query requests) from the MINE server, IPNE decodes it and sends the corresponding clp handle to the session manager. The information that is shared is a snapshot of the session/flow/nbr context at that instance in time.

Architecture

The MINE IPNE client is implemented as a configurable service on P-GW, HA, GGSN or HNBGW services as illustrated below.

Figure 2: Detailed IPNE Architecture



Limitations

Note the following limitations for the IPNE feature:

- The IPNE service implements a flow control mechanism over the XMPP interface. As a result, any messaging over this interface which exceeds the set queue thresholds would be discarded.

Flows

This section provides call flow diagrams for IPNE Query, Subscription, Feed, Addition and Deletion scenarios. Some flow diagrams use the P-GW as an example, but they also apply to GGSN, HA, and HNBGW as well.

Figure 3: IPNE Handling of Query from MINE Server

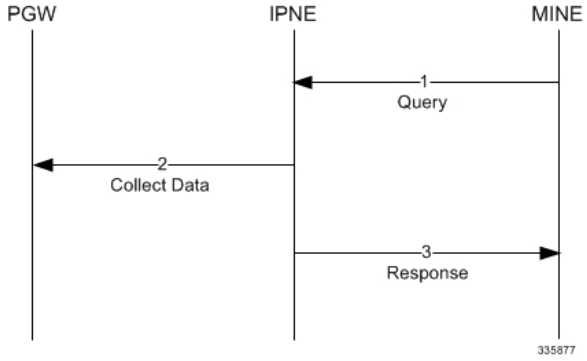


Table 1: IPNE Handling of Query from MINE Server

Step	Description
1	The MINE server sends a query over the XMPP stream to the IPNE service. The query is XML encoded, which contains a query-id, key to look up a session (for example, sessmgr instance:callid), and a list of segments specifying the interested information.
2	Upon receipt of the query, the IPNE service parses the XML data and finds the handle using the key provided by MINE server, and then invokes the registered call back function to collect the session information. The requested information is also provided to the call back function in the form of a bit mask.
3	With the help of XML encoder, the IPNE service converts the session information to XML format and sends it to the MINE server.

Figure 4: IPNE Service Handling a Subscription from the MINE Server

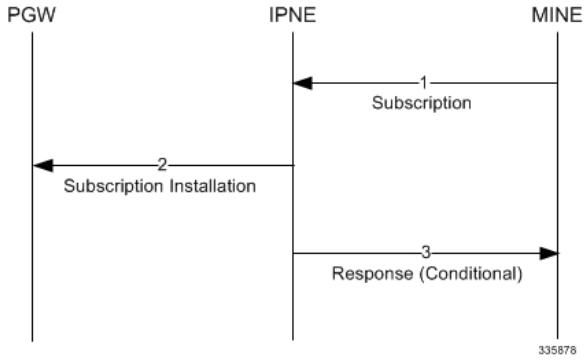


Table 2: IPNE Service Handling a Subscription from the MINE Server

Step	Description
1	The MINE server sends a subscription over the XMPP stream to the IPNE service. The subscription is XML-encoded and has a similar format as the query message, e.g. a list of fragments specifying the feed triggers.
2	The subscription installation is maintained by the IPNE on a per handle basis.
3	This step is conditional. If there are any existing sessions that match any of the triggers listed in the subscription, A success acknowledgement message is sent to the MINE server.

Figure 5: IPNE Service Sends Unsolicited Feed Message to the MINE Server

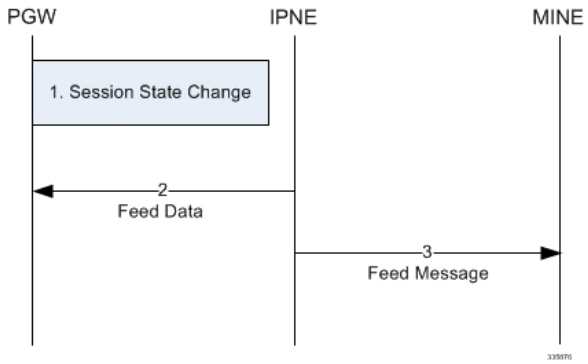


Table 3: IPNE Service Sends Unsolicited Feed Message to the MINE Server

Step	Description
1	A session detects some state change, for example, a RAT change due to a handoff.
2	The session invokes a public API on the handle to inform the IPNE service of the change.
3	If the change matches any of the subscription installations installed on the IPNE handle, a feed message is built and sent to the MINE server(s).

Figure 6: IPNE Session Addition

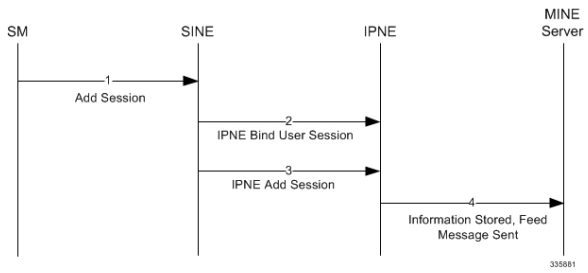


Table 4: IPNE Session Addition

Step	Description
1	While setting up the session, the session manager application checks to see if IPNE is enabled.
2	If IPNE is enables, the SM sends the add session information to the SINE interface.
3	SINE binds the session information and sends the add event towards the IPNE application.
4	Information is stored, and the information is passed as a feed message to the MINE server in the form of XML data.

Figure 7: IPNE Session Deletion

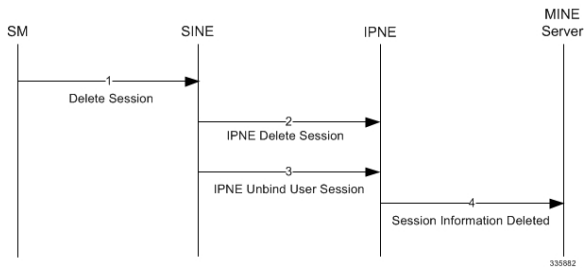


Table 5: IPNE Session Deletion

Step	Description
1	Before releasing the session, the session manager application calls for delete_session.
2	SINE invokes the delete session event.
3	SINE unbinds the session information.
4	The corresponding session information is deleted from the IPNE application and a feed message is sent to the MINE server with type as delete.

Standards Compliance

The StarOS IPNE feature complies with the following standards:

- RFC 6120; Extensible Messaging and Presence Protocol (XMPP): Core, Section 4.7 (Stream attributes)

Configuring the IPNE Feature

This section describes how to configure the IPNE feature and how to verify the configuration.

Configuring IPNE

This section describes how to configure the IPNE feature.

Configuring IPNE includes configuring the IPNE service, and then associating the IPNE service with the GGSN, HA, HNBGW or P-GW service.

Use the following example configuration to create the IPNE service.

```
config
 context context_name
 ipne ipne_service_name
 ipne-endpoint
 bind [ ipv4 ipv4_address | ipv6 ipv6_address ]
 peer [ ipv4 | ipv6 ] protocol tcp
 end
```

Notes:

- Both the **bind** and **peer** keywords support IPv4 and IPv6 addressing.
- **tcp** is the default transport protocol. SCTP is not supported at this time.
- The default XMPP protocol port is 5222.
- The **fqdn**, **priority** and **weight** keywords are not supported at this time.

HNBGW only. Usually, notify messages are sent only on subscription. However, an exception has been made for HNBGW UE Registration / Deregistration. HNBGW UE Registration/Deregistration will always be notified without any subscription. To control the sending of such unsolicited notification, enter the following command:

```
configure
      context  ipne_service_name
            unsolicited-notify-trigger hnb-ue
            end
```

Notes

- If **unsolicited-notify-trigger hnb-ue** is configured, the IPNE service sends notifications for UE Register/De-register requests on receiving the requests from the HNBGW.
- If **no ununsolicited-notify-trigger hnb-ue** is configured, the IPNE will not send UE Register/De-register notifications. This is the default setting.

Once the IPNE service has been created, it must be associated with the configured GGSN, HA, P-GW or HNBGW service. Use the following example to associate the IPNE service with the configured gateways service

```
configure
      context  gw_context_name
 associate ipne-service  ipne_service_name
           end
```

Notes:

- **context** *gw_context_name* is the name of the configured GGSN, HA, P-GW or HNBGW service name configured on the StarOS
- To remove the association between the IPNE service and the gateway service, use the **no associate ipne-service**command.

Verifying the IPNE Configuration

This section describes how to verify the IPNE configuration

From exec mode issue the following command to verify the IPNE configuration:

```
show ipne peers all
```

The output of this command provides the following information for each IPNE service instance:

- IPNE Service Name
- Context ID
- Peer IP address
- State of the TCP connections to the peer.

Monitoring the IPNE Service

This section describes how to monitor the StarOS IPNE feature.

IPNE Show Commands

This section provides information regarding show commands and/or their outputs in support of the StarOS IPNE feature.

The show commands in this section are available in support of the the StarOS IPNE feature.

show ipne peers all

This command provides a list of peers of each IPNE service and the state of the TCP connections.

show ipne statistics all

This command shows the total number of handles for each IPNE service and counter totals for queries, responses, subscriptions and feeds.

show active-charging subscsribers full all

This command shows if the MINE server has currently subscribed notifications for this ACS session or not (**IPNE enabled** or **disabled**). It also indicates the number of notifications sent to the MINE server for this

ACS session. Historical notification counts across all current and deleted flows are stored. If the MINE server has not been subscribed for notifications, this field reads **n/a**.