NRF Logging

Feature Summary and Revision History

Summary Data

Table 1: Summary Data

Applicable Product(s) or Functional Area	5G-NRF
Applicable Platform(s)	SMI
Feature Default Setting	Enabled - Always-on
Related Changes in this Release	Not Applicable
Related Documentation	Not Applicable

Revision History

Table 2: Revision History

Revision Details	Release
First introduced.	2026.01

Feature Description

NRF utilizes the common logging framework to generate logs from its micro-services.

The supported log levels are:

- Error
- Warn
- Info
- Debug
- Trace

Note Warn level logging takes place during production.

Error

These errors are fatal errors, which can impact service for multiple subscribers. Examples errors are as follows:

- Node discovery of SBA fails after query from NRF and local configuration.
- Mandatory IE missing in NF subscription request.
- Memory cache startup errors.
- Endpoint not found.

Warn

These errors impact few specific call-flows majorly, but not blockers of functionality. Example errors are as follows:

- Node discovery of SBA fails but we have more options to retry.
- Mandatory IE missing in a registration message.
- RPC timeout.
- Procedural timeout.
- Validation failure (not critical). Example: Registration rejected as registration request message received registration type as not supported registration type.
- External entity sending unexpected or negative response. Example: Heart beat failure.
- Unexpected value of objects maintained by NRF.
- Unable to fetch a subscription.

Info

This log level purpose is to know information for cause. Examples are as follows:

- Procedural outcome. Example:NF Registration.
- IE changes during NF Update.

Debug

This log level purpose is to get debug messages. Example messages are as follows:

- All external exchanged messages. Example: Sending Registration request.
- State machine changes.
- Collision detailed logging.

Trace

This log level purpose is to get content of all external tracing messages. Example messages are as follows:

- Registration request message.
- Subscription requests and notifications.

How It Works

It is achieved using Log tags and Logging contexts.

Log Tags

Use Log tags to 'tag' logs for specific procedures which are part of flow or an event. Enabling of NRF logging takes place at different log levels for different log tags.

Name	Purpose	Example Log tags
NRF Service	To capture procedures.	• LogTagReg • LogTagNrfDeReg • LogTagNrfDis • LogTagNrfNotify
Rest EP	To capture on the interface basis.	• LogTagReg • LogTagNrfDeReg • LogTagNrfDis • LogTagNrfNotify

Logging contexts

App Infra context (Transaction)

Capturing of transaction logging is to get the summary of a transaction when it ends. This transaction captures log tag without the filename or a line number.

Transaction logging captures information at the Error and Warn levels.

CLI option is available to duplicate transaction logs, which capture transaction logs inline too.

```
************ TRANSACTION: 00005 *************
TRANSACTION SUCCESS:
Start Time : 2022/04/20 11:33:55.165
GR Instance ID : 1
Txn Type : NFRegistrationRequest(1)
Priority : 0
Session Namespace : none(0)
CDL Slice Name : slice1
Subscriber Id : 123c2c0b-4bf6-4204-8225-2aa2a9abc100
LOG MESSAGES:
2022/04/20 11:33:55.171 [DEBUG] [infra.ipc_action.core] Destination Host
NRF.nrf-service.blr.nrf.0 serviced the IPC Message NFRegistrationRequest
*********************************************

2022/04/20 11:33:55.170 nrf-rest-ep [DEBUG] [IpcStreamClient.go:314] [infra.bgipcstream.core]
 Received Response &ResponseMessage{Key:1,Code:0,MsgType:2,Description:,Message:[10 36 49
50 51 99 50
99 48 98 45 52 98 102 54 45 52 50 48 52 45 56 50 50 53 45 50 97 97 50 97 57 97 98 99 49 48
 48 16 1 26 226 1 42 51 10 2 65 65 18 3 48 65 65 42 19 10 6 97 98 99 100 50 51 18 9 10 3
49 50 51 18 2 52
53 42 19 10 6 65 66 48 48 48 49 18 9 10 3 49 50 51 18 2 52 53 64 216 54 104 200 1 122 14
49 57 50 46 49 54 56 46 49 50 48 46 50 53 122 13 49 57 50 46 49 54 56 46 49 48 48 46 51 130
 1 20 49 49 49
49 58 58 49 57 50 58 49 54 56 58 49 50 48 58 50 53 130 1 19 49 49 49 49 58 58 49 57 50 58
49 54 56 58 49 48 48 58 51 136 1 20 146 1 8 66 97 110 103 103 103 103 103 154 1 36 49 50
51 99 50 99 48 98
 45 52 98 102 54 45 52 50 48 52 45 56 50 50 53 45 50 97 97 50 97 57 97 98 99 49 48 48 194
1 10 82 69 71 73 83 84 69 82 69 68 202 1 3 65 77 70 242 1 9 10 3 49 50 51 18 2 52 53 248 1
 10 176 2 241 3],
Name:,DestName:NRF.nrf-service.blr.nrf.0,MetaInfo:&MetaInfo{Priority:0,InstanceInfo:
NRF.nrf-service.blr.nrf.0,SubscriberID:,SlaEnabled:false,SlaTimeoutInMs:0,ProcedureInfo:
&ProcedureInfo{ProcedureName:,
SubProcedureName:,ProtocolStartTimestamp:4751572391071609,ServiceStartTimestamp:0,
ProcedureStage:0,IpcReqStartTimestamp:4751572392168693,IpcRespStartTimestamp:
4751572392925193,},CorrelationId:,
TraceInfo:nil,RpcInfo:[]*RpcInfo{},ExternalMsg:false,GRInstanceID:1,TestCaseID:0,
ProtocolPayloadLength:0,},RoutingRules:map[string]string{InstanceInfo:
NRF.nrf-service.blr.nrf.0,SubscriberID:
123c2c0b-4bf6-4204-8225-2aa2a9abc100,},} at client nrf-service_0
2022/04/20 11:33:55.171 nrf-rest-ep [DEBUG] [IpcStreamClient.go:325] [infra.bgipcstream.core]
 Response object &{%!s(chan *ipc.ResponseMessage=0xc003b0cf60) %!s(bool=false) Sync} at
client nrf-service_0
2022/04/20 11:33:55.171 nrf-rest-ep [DEBUG] [IpcStreamAction_private.go:137]
[infra.ipc_action.core] Time taken to execute IPC is 0.00
2022/04/20 11:33:55.171 nrf-rest-ep [DEBUG] [MasterBlueprint.go:516] [infra.transaction.core]
 Last stage ( init_done ) -> Next stage ( finished ) for transaction : 5
2022/04/20 11:33:55.171 nrf-rest-ep [TRACE] [rest_message_processor.go:144]
[rest_ep.msg-process.Int] [5]In ProcessEnd
2022/04/20 11:33:55.171 nrf-rest-ep [TRACE] [rest_message_processor.go:174]
[rest_ep.msg-process.Int] [5]In PopulateSuccessResponse
2022/04/20 11:33:55.171 nrf-rest-ep [INFO] [register_nf_request.go:1935] [rest_ep.app.Reg]
```

```
 [123c2c0b-4bf6-4204-8225-2aa2a9abc100]Sending Successful Response for NF Registration
Request
```