|  |  |
| --- | --- |
| 3GPP TR 29.808 V16.0.0 (2019-12) | |
| Technical Report | |
| 3rd Generation Partnership Project;  Technical Specification Group Core Network and Terminals;  Study on the Nudsf Service Based Interface;  (Release 16) | |
|  | |
|  |  |
|  | |
| The present document has been developed within the 3rd Generation Partnership Project (3GPP TM) and may be further elaborated for the purposes of 3GPP. The present document has not been subject to any approval process by the 3GPPOrganizational Partners and shall not be implemented. This Specification is provided for future development work within 3GPPonly. The Organizational Partners accept no liability for any use of this Specification. Specifications and Reports for implementation of the 3GPP TM system should be obtained via the 3GPP Organizational Partners' Publications Offices. | |

|  |
| --- |
|  |
| ***3GPP***  Postal address  3GPP support office address  650 Route des Lucioles - Sophia Antipolis  Valbonne - FRANCE  Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16  Internet  http://www.3gpp.org |
| ***Copyright Notification***  No part may be reproduced except as authorized by written permission. The copyright and the foregoing restriction extend to reproduction in all media.  © 2019, 3GPP Organizational Partners (ARIB, ATIS, CCSA, ETSI, TSDSI, TTA, TTC).  All rights reserved.  UMTS™ is a Trade Mark of ETSI registered for the benefit of its members  3GPP™ is a Trade Mark of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners LTE™ is a Trade Mark of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners  GSM® and the GSM logo are registered and owned by the GSM Association |

Contents

Foreword 5

1 Scope 6

2 References 6

3 Definitions of terms, symbols and abbreviations 6

3.1 Terms 6

3.2 Symbols 6

3.3 Abbreviations 7

4 Architecture and Requirements 7

4.1 Reference Architecture 7

4.1.1 General 7

4.1.2 UDSF Services 7

4.2 General Requirements 8

5 Solutions for the Nudsf SBI 9

5.1 Introduction 9

5.2 Solution #1 – SBI based solution with Key-Value Pair 9

5.2.1 Introduction 9

5.2.1.1 General 9

5.2.2 Resources 10

5.2.2.1 Resource Structure 10

5.2.2.2 Realms 10

5.2.2.2.1 Description 10

5.2.2.2.2 Operations 10

5.2.2.3 Storages 10

5.2.2.3.1 Description 10

5.2.2.3.2 Operations 10

5.2.2.4 RecordReference 11

5.2.2.4.1 Description 11

5.2.2.4.2 Operations 11

5.2.2.5 Record 11

5.2.2.5.1 Description 11

5.2.2.5.2 Operations 11

5.2.2.6 Blocks 11

5.2.2.6.1 Description 11

5.2.2.6.2 Operations 11

5.2.2.7 Block 12

5.2.2.7.1 Description 12

5.2.2.7.2 Operations 12

5.2.2.8 Meta 12

5.2.2.8.1 Description 12

5.2.2.8.2 Operations 12

5.2.2.9 Subscriptions Collection 12

5.2.2.9.1 Description 12

5.2.2.9.2 Operations 12

5.2.2.10 Subscription 12

5.2.2.10.1 Description 12

5.2.2.10.2 Operations 13

5.2.3 Latency and Performance Analysis 13

5.2.3.1 Introduction 13

5.2.3.2 Deployment Topology 13

5.2.3.3 Caching 13

5.2.3.4 Parallel Access 13

5.2.3.5 Database Schema 13

5.2.3.6 HTTP/2 Protocol 14

5.2.3.7 Conclusion 14

5.3 Evaluation and Conclusion 14

Annex A (informative): Change history 16

# Foreword

This Technical Report has been produced by the 3rd Generation Partnership Project (3GPP).

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

x the first digit:

1 presented to TSG for information;

2 presented to TSG for approval;

3 or greater indicates TSG approved document under change control.

y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.

z the third digit is incremented when editorial only changes have been incorporated in the document.

In the present document, modal verbs have the following meanings:

**shall** indicates a mandatory requirement to do something

**shall not** indicates an interdiction (prohibition) to do something

The constructions "shall" and "shall not" are confined to the context of normative provisions, and do not appear in Technical Reports.

The constructions "must" and "must not" are not used as substitutes for "shall" and "shall not". Their use is avoided insofar as possible, and they are not used in a normative context except in a direct citation from an external, referenced, non-3GPP document, or so as to maintain continuity of style when extending or modifying the provisions of such a referenced document.

**should** indicates a recommendation to do something

**should not** indicates a recommendation not to do something

**may** indicates permission to do something

**need not** indicates permission not to do something

The construction "may not" is ambiguous and is not used in normative elements. The unambiguous constructions "might not" or "shall not" are used instead, depending upon the meaning intended.

**can** indicates that something is possible

**cannot** indicates that something is impossible

The constructions "can" and "cannot" are not substitutes for "may" and "need not".

**will** indicates that something is certain or expected to happen as a result of action taken by an agency the behaviour of which is outside the scope of the present document

**will not** indicates that something is certain or expected not to happen as a result of action taken by an agency the behaviour of which is outside the scope of the present document

**might** indicates a likelihood that something will happen as a result of action taken by some agency the behaviour of which is outside the scope of the present document

**might not** indicates a likelihood that something will not happen as a result of action taken by some agency the behaviour of which is outside the scope of the present document

In addition:

**is** (or any other verb in the indicative mood) indicates a statement of fact

**is not** (or any other negative verb in the indicative mood) indicates a statement of fact

The constructions "is" and "is not" do not indicate requirements.

# 1 Scope

The present document provides a study of a service based interface to the UDSF. It also evaluates the latency/performance of an SBI based Nudsf and non-SBI based alternatives (if any).

# 2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non‑specific.

- For a specific reference, subsequent revisions do not apply.

- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document*.

[1] 3GPP TR 21.905: "Vocabulary for 3GPP Specifications".

[2] 3GPP TS 23.501: "System Architecture for the 5G System; Stage 2".

[3] 3GPP TS 23.502: "Procedures for the 5G System; Stage 2".

[4] 3GPP TS 29.501: "Principles and Guidelines for Services Definition".

[5] 3GPP TS 29.500: "Technical Realization of Service Based Architecture".

# 3 Definitions of terms, symbols and abbreviations

## 3.1 Terms

Void.

## 3.2 Symbols

Void.

## 3.3 Abbreviations

For the purposes of the present document, the abbreviations given in 3GPP TR 21.905 [1] and the following apply. An abbreviation defined in the present document takes precedence over the definition of the same abbreviation, if any, in 3GPP TR 21.905 [1].

KVP Key-Value Pair

NF Network Function

UDSF Unstructured Data Storage Function

UDR Unified Data Repository

# 4 Architecture and Requirements

## 4.1 Reference Architecture

### 4.1.1 General

The Data Storage architectures are defined in clause 4.2.5 of 3GPP TS 23.501 [2]:

As depicted in Figure 4.1.1-1, the 5G System architecture allows any NF to store and retrieve its unstructured data into/from a UDSF (e.g. UE contexts). The UDSF belongs to the same PLMN where the network function is located. CP NFs may share a UDSF for storing their respective unstructured data or may each have their own UDSF (e.g. a UDSF may be located close to the respective NF).

NOTE 1: Structured data in this specification refers to data for which the structure is defined in 3GPP specifications. Unstructured data refers to data for which the structure is not defined in 3GPP specifications.



Figure 4.1.1-1: Data storage architecture for unstructured data from any NF

In addition, clause 6.2.12 of 3GPP TS 23.501 [2] indicates:

NOTE 2: Deployments can choose to collocate UDSF with UDR.

### 4.1.2 UDSF Services

The UDSF Services are defined in clause 5.2.14 of 3GPP TS 23.502 [3]:

The following table illustrates the UDSF Services.

Table 4.1.2-1: NF Services provided by UDSF

|  |  |  |  |
| --- | --- | --- | --- |
| NF service | Service Operations | Operation Semantics | Example Consumer(s) |
| Unstructured | Query | Request/Response | Any NF |
| Data | Create | Request/Response | Any NF |
| Management | Delete | Request/Response | Any NF |
|  | Update | Request/Response | Any NF |

## 4.2 General Requirements

The requirements to the UDSF function are described below.

- Data management requirements:

- The NF shall be able to store and retrieve the data with the following basic operations:

- Create data.

- Read data.

- Update data.

- Delete data.

- In addition the solution should support the following capabilities:

- Transactional integrity.

- Pessimistic and or optimistic locking.

- If multiple instances of the same NF type and same vendor can read and write the same data, the following additional operations may be considered:

- Subscribe to notifications of data change.

- Notification of a data change.

- Performance requirements:

- The UDSF may be used to store data with very different characteristics and with different real time performance requirements ranging from low to very high ones.

- The protocol used over N18 shall provide latency as low as possible.

- Multiple logical storage spaces:

- NFs of the same type and same vendor may need to store different data in different repositories. The UDSF shall support multiple logical storage spaces. Each logical storage unit shall have a unique identifier.

- UDSF sharing:

- NFs may share a UDSF for storing their respective unstructured data. NFs of the same type and vendor shall have one or more different logical storage spaces and it shall be possible to prevent NFs of another type or another vendor from accessing it. NFs of different types or of different vendors shall use different access keys.

- Collocation with UDR:

- UDSF may be collocated with the UDR.

- Load and overload requirement:

- The solution shall support load control mechanisms to allow an automatic distribution of the traffic load amongst the different instances of the UDSF.

- The solution shall support overload control mechanisms to protect UDSF instance when they reach a congestion state and to request the NFs to throttle the requests sent to the UDSF.

- The usage of intermediaries (e.g. SCP) for these load and overload control purposes shall also be evaluated.

- Security requirements:

- Transport of messages between the NFs and UDSF should be protected to provide privacy and data integrity. Client applications should be authenticated.

- NF independent:

- The architecture is applicable to any NF. The solution shall be independent from the NF type.

- Types of Data:

- In addition to common data types, the UDSF shall support binary type of data.

# 5 Solutions for the Nudsf SBI

## 5.1 Introduction

This clause describes the solution(s) for the Nudsf SBI.

## 5.2 Solution #1 – SBI based solution with Key-Value Pair

### 5.2.1 Introduction

#### 5.2.1.1 General

The service based Nudsf in this solution follows the principles for a 5GC SBI API as defined in 3GPP TS 29.501 [4].

The solution utilizes key-value pair (KVP) principles and provides a flexible structure for partitioning the space within the UDSF so that a multitude of applications and types of data can be managed independently with minimum risk of e.g. namespace collisions.

The Nudsf storage model relies on structural abstractions with elements referred to as Realms, Storages, Record and Block.



Figure 5.2.1.1-1: Nudsf storage model example

### 5.2.2 Resources

#### 5.2.2.1 Resource Structure



Figure 5.2.2.1-1: Resource URI structure of the Nudsf-dr API

#### 5.2.2.2 Realms

##### 5.2.2.2.1 Description

A realm is an area or a domain of the UDSF; a storage location where the actual underlying data objects can be organized.

The realm provides a layer of abstraction allowing for physical and logical data segregation if needed. An application could utilize one or more realms for its data. Two applications could share the same realm in order to share common data.

The realm is identified with a unique realmId and is of the archetype Collection.

##### 5.2.2.2.2 Operations

The Realms resource does not support any operations, i.e. realms are created in the UDSF through the means of O&M operations.

#### 5.2.2.3 Storages

##### 5.2.2.3.1 Description

A storage defines a data model within the Realm for a specific application or purpose.

The storage is used to store records of the same or similar type and is identified with a storageId and is of the archetype Collection.

##### 5.2.2.3.2 Operations

The Storages resource does not support any operations, i.e. storages are created in the UDSF through the means of O&M operations.

#### 5.2.2.4 RecordReference

##### 5.2.2.4.1 Description

A RecordReference represents the collection of records within a storage. It can be used to search for specific records matching certain criteria.

The RecordReference is of the archetype Collection.

##### 5.2.2.4.2 Operations

The RecordReference will support the following standard operations:

- GET: retrieves one or more records.

The GET operation returns the (possible empty) list of references to records within the collection matching the specified filter criteria. The criteria is based on record tags in the associated meta resource. More specifically, it searches for records having in the metadata a tag matching the conditional expression set in the filter criteria definition.

The search criteria for EQ a meta tag and a value is mandatory to support. Optionally NEQ, GT, GTE, LT, LTE, and search conditions, e.g., AND, OR, and NOT may be supported. It is optional for a tag within the meta resource to have a unique value.

Since the search can return a large set of matching record references, a "limit-range" (max number of records returned) and "page-number" (the first record to be returned) can be indicated.

#### 5.2.2.5 Record

##### 5.2.2.5.1 Description

A record is comprised of a single key-value pair (KVP) where the key, which is a unique identifier for some item of data, and the value, which is the data that is identified by the key.

The Record is an abstract data type that includes a key identifier and a set of associated values (blocks).

The Record is identified with a recordId and is of the archetype Document.

##### 5.2.2.5.2 Operations

The Record will support the following standard operations:

- GET: retrieves the list of blocks and meta (if present) associated with the recordId.

- PUT: create the recordId and associated block(s) (if any).

- DELETE: delete the record and any associated blocks.

- HEAD: check record existence.

#### 5.2.2.6 Blocks

##### 5.2.2.6.1 Description

The blocks resource supports the retrieval of all blocks associated with the record.

The blocks resource is of the archetype Collection.

##### 5.2.2.6.2 Operations

- GET: retrieves all blocks associated with the recordId.

#### 5.2.2.7 Block

##### 5.2.2.7.1 Description

The Block represents the value part of the Record and is made up of opaque arbitrary content.

The content part can be of a simple data type, structured data type, array, map, enumeration, data type describing alternative data types, data type describing combinations of data types or "Any Type".

The Block is identified with a blockId and is of the archetype Document.

##### 5.2.2.7.2 Operations

- GET: retrieves the block keyed with the blockId.

- PUT: creates a block with blockId if blockId doesn't exist; otherwise updates the block associated with the blockId.

- DELETE: delete the block associated with blockId.

- HEAD: check block existence.

#### 5.2.2.8 Meta

##### 5.2.2.8.1 Description

The Meta is an optional resource that supports the storage of tags and values associated with the record.

Tags and values are described in a JSON representation and are populated by the NF service consumer. The meta is used in GET operations to allow filtering of records.

The Meta is of the archetype Document.

##### 5.2.2.8.2 Operations

- GET: retrieves a record meta data.

- PATCH: partial update of the record meta data.

#### 5.2.2.9 Subscriptions Collection

##### 5.2.2.9.1 Description

The Subscriptions Collection represents the collection of subscriptions within a storage for triggering notifications.

The Subscriptions Collection is of the archetype Collection.

NOTE: The subscription to notifications is an optional capability.

##### 5.2.2.9.2 Operations

- POST: Creates a subscription.

#### 5.2.2.10 Subscription

##### 5.2.2.10.1 Description

The Subscription resource represents a subscription for triggering notifications on storage records.

The Subscription is of the archetype Document.

##### 5.2.2.10.2 Operations

- GET: Retrieves the subscription keyed by the SubsId.

- PUT: Updates the subscription for the supplied SubsId. If a subscription does not exist, the request will fail.

- DELETE: Delete the subscription associated with the supplied SubsId.

- HEAD: Check subscription existence.

### 5.2.3 Latency and Performance Analysis

#### 5.2.3.1 Introduction

For deployments where the UDSF stores state information for the purpose of stateless deployments of e.g. AMF/SMF/PCF NFs, the transaction rate towards an UDSF instance can be significant. Further, to minimize core network latency, the response time must be kept at a minimum.

In general, response time can be expressed as:

Wait Time = Network Roundtrip Latency + Wait in Queue + Service (processing) Time.

To reduce the wait time, the network latency, the wait in queue time and the service time all play a role in the overall perceived latency.

The following clauses describes techniques that allows for a high throughput and low latency when a UDSF is deployed with a service based interface.

#### 5.2.3.2 Deployment Topology

The UDSF instance should be close to the NF service consumer, i.e. in the same data center to allow for maximum bandwidth and to reduce the Network Roundtrip Latency to a minimum.

A cloud native deployment of the UDSF allows for distributing the function across many compute nodes and thus allows for massive parallelism in processing requests.

#### 5.2.3.3 Caching

A stateless NF service consumer can maintain a local cache which will reduce the requests to the UDSF and the response time significantly.

If the NF service consumer is mainly exporting state info for the purpose of allowing another NF service consumer instance to take over in case of an outage of the original NF service consumer instance, there is little risk that the cache is outdated if all transactions for a given UE are processed by a single NF service consumer instance.

#### 5.2.3.4 Parallel Access

If an NF service consumer is representing its state info in multiple resources, it may request multiple resources at the same time from the UDSF, as opposed to requesting them one at a time. The stream concept in HTTP/2 allows for multiplexed messages through a single connection. When the UDSF is deployed in a cloud native distributed cluster, a single UDSF instance has capacity to handle a large number of transactions in parallel and thus the Wait time for two or more requests may not be much greater than the wait time of a single request.

#### 5.2.3.5 Database Schema

This solution allows the "value" part of the KVP to be represented with multiple blocks allowing the NF service consumer designer to design a data model that optimizes Read vs Update info.

For example, if an NF service consumer stores 80kB of state info, it may periodically have a need to update e.g. a time stamp that is e.g. only 4 bytes in length. Using a PUT with the complete updated payload of 80k vs 4 bytes is obviously significantly more expensive from a consumption of network and compute node resources.

#### 5.2.3.6 HTTP/2 Protocol

HTTP/2 supports a number of performance improving (compared to HTTP/1.1) capabilities, such as:

- Interleave of multiple requests in parallel.

- Interleave of multiple responses in parallel.

- Use of a single connection to deliver multiple requests and responses in parallel.

- Deliver lower page load times by eliminating unnecessary latency and improving utilization of available network capacity.

- Optimization with request and response header compression (HPACK).

#### 5.2.3.7 Conclusion

As described in this clause, an SBI based UDSF can support high throughput and low latency when architected and deployed in an optimal manner.

## 5.3 Evaluation and Conclusion

The following table provides the analysis of the requirements from clause 4.2 with the solution described in clause 5.2:

Table 5.3-1: Requirements Analysis of the Solution

|  |  |
| --- | --- |
| Requirement | Analysis |
| Data management requirements:  The NF shall be able to store and retrieve the data with the following basic operations:  - Create data.  - Read data.  - Update data.  - Delete data.  In addition the solution should support the following capabilities:  - Transactional integrity.  - Pessimistic and or optimistic locking. | The proposed solution supports the basic operations.  Transactional Integrity is the responsibility of the UDSF vendor.  The optimistic locking mechanism using the eTag header as described in 3GPP TS 29.500 [5] applies to the Nudsf interface. |
| If multiple instances of the same NF type and same vendor can read and write the same data, the following additional operations may be considered:  - Subscribe to notifications of data change.  - Notification of a data change. | The solution describes a Subscribe to Notify mechanism. |
| Performance requirements:  The UDSF may be used to store data with very different characteristics and with different real time performance requirements ranging from low to very high ones.  The protocol used over N18 shall provide latency as low as possible. | Clause 5.2.3 describes solutions on how this can be accomplished. |
| Multiple logical storage spaces:  NFs of the same type and same vendor may need to store different data in different repositories. The UDSF shall support multiple logical storage spaces. Each logical storage unit shall have a unique identifier. | The solution supports the concept of a UDSF supporting one or more realms. Within a realm, one or more storages are supported. |
| UDSF sharing:  NFs may share a UDSF for storing their respective unstructured data. NFs of the same type and vendor shall have one or more different logical storage spaces and it shall be possible to prevent NFs of another type or another vendor from accessing it. NFs of different types or of different vendors shall use different access keys. | The solution supports the concept of a UDSF supporting one or more realms. Within a realm, one or more storages are supported.  Access control is outside the scope of standardization, but the current resource structure allows for it. |
| Collocation with UDR:  UDSF may be collocated with the UDR. | An SBI based UDSF is ideal to be co-located with the UDR. |
| Load and overload requirement:  The solution shall support load control mechanisms to allow an automatic distribution of the traffic load amongst the different instances of the UDSF.  The solution shall support overload control mechanisms to protect UDSF instance when they reach a congestion state and to request the NFs to throttle the requests sent to the UDSF.  The usage of intermediaries (e.g. SCP) for these load and overload control purposes shall also be evaluated. | The rel-16 work item on "Load and Overload Control of 5GC Service Based Interfaces" applies to the Nudsf interface. |
| Security requirements:  Transport of messages between the NFs and UDSF should be protected to provide privacy and data integrity. Client applications should be authenticated. | The usage of TLS for transport security as described in 3GPP TS 29.500 [5] applies to the Nudsf.  The usage of OAuth2 for Authorization of NF service access as described in 3GPP TS 29.500 [5] applies to the Nudsf. |
| NF independent:  The architecture is applicable to any NF. The solution shall be independent from the NF type. | The proposed solution applies to any NF of any NF type. |
| Types of Data:  In addition to common data types, the UDSF shall support binary type of data. | Binary data is supported by the proposed solution. |

The solution as proposed in clause 5.2 is well suited for a 3GPP defined Nudsf interface and shall be the basis for a normative Nudsf interface.

Annex A (informative):  
Change history

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Change history** | | | | | | | |
| **Date** | **Meeting** | **TDoc** | **CR** | **Rev** | **Cat** | **Subject/Comment** | **New version** |
| 2019-04 | CT4#90 | C4-191434 |  |  |  | Skeleton of the TR | 0.0.1 |
| 2019-05 | CT4#91 | C4-192487 |  |  |  | Implementation of pCRs agreed in CT4#91 – C4-192408, C4-192409 and C4-192410 | 0.1.0 |
| 2019-08 | CT4#93 | C4-193262 |  |  |  | Implementation of pCR C4-193262 | 0.2.0 |
| 2019-10 | CT4#94 | C4-194522 |  |  |  | Implementation of pCRs C4-194271, C4-194272, C4-194273, C4-194351. | 0.3.0 |
| 2019-11 | CT4#95 | C4-195637 |  |  |  | Implementation of pCRs C4-195013, C4-195015, C4-195486 and C4-195488 | 0.4.0 |
| 2019-12 | CT#86 | CP-193068 |  |  |  | TR presented for information and approval | 1.0.0 |
| 2019-12 | CT#86 |  |  |  |  | TR approved | 16.0.0 |