Security analysis of DCCF analytics availability and fetch

This report in another language: 中文.

19.7.0 · ProVerif+ Agent report

Documents	5G System; Data Collection Coordination Services; Stage 3 (protocol definition); 5G System; Network Data Analytics Services; Stage 3 (supporting)
Properties examined	11: 10 hold, 1 fail (1 attack)
Findings	1

Contents

1 Protocol and scope
2 Findings
3 Properties examined
4 Limits of this analysis
Appendix A: Reading of each property
Appendix C: Claims and prover answers
Appendix D: How the analysis was run

1 Protocol and scope

The Ndccf_DataManagement API of TS 29.520 lets an NF service consumer subscribe at the DCCF for aggregated analytics that the DCCF collects from an NWDAF. The flow has three parties: the NF service consumer, which holds its OAuth2 access token and its TLS credentials, the DCCF, which holds the stored subscriptions and the analytics it collected, and the NWDAF, which produces the analytics.

The consumer subscribes and names a notification address, anaNotifUri, and a notification correlation identifier, anaNotifCorrId. The DCCF answers the subscription, subscribes upstream, and when the analytics is ready it notifies the consumer and gives it a fetch instruction instead of the data, made of fetchUri and fetchCorrIds. The consumer acknowledges the notification and then posts the fetch correlation identifiers to the fetch URI to collect the analytics.

When the exchange ends the consumer holds analytics that the DCCF received from the NWDAF on the notification path of its own subscription, tied to its own correlation identifiers. The DCCF holds a subscription whose contents the authenticated consumer proposed, and answers a Fetch only for identifiers it issued and only to the consumer that presents them.

The parties:

- **NF service consumer**: subscribes for analytics and collects them. Holds: OAuth2 access token, TLS credentials.
- **DCCF**: coordinates the collection and notifies availability. Holds: stored subscriptions, issued fetchCorrIds, stored analytics.
- **NWDAF**: produces the analytics. Holds: analytics source.

Figure 1 shows the exchange as the documents specify it.

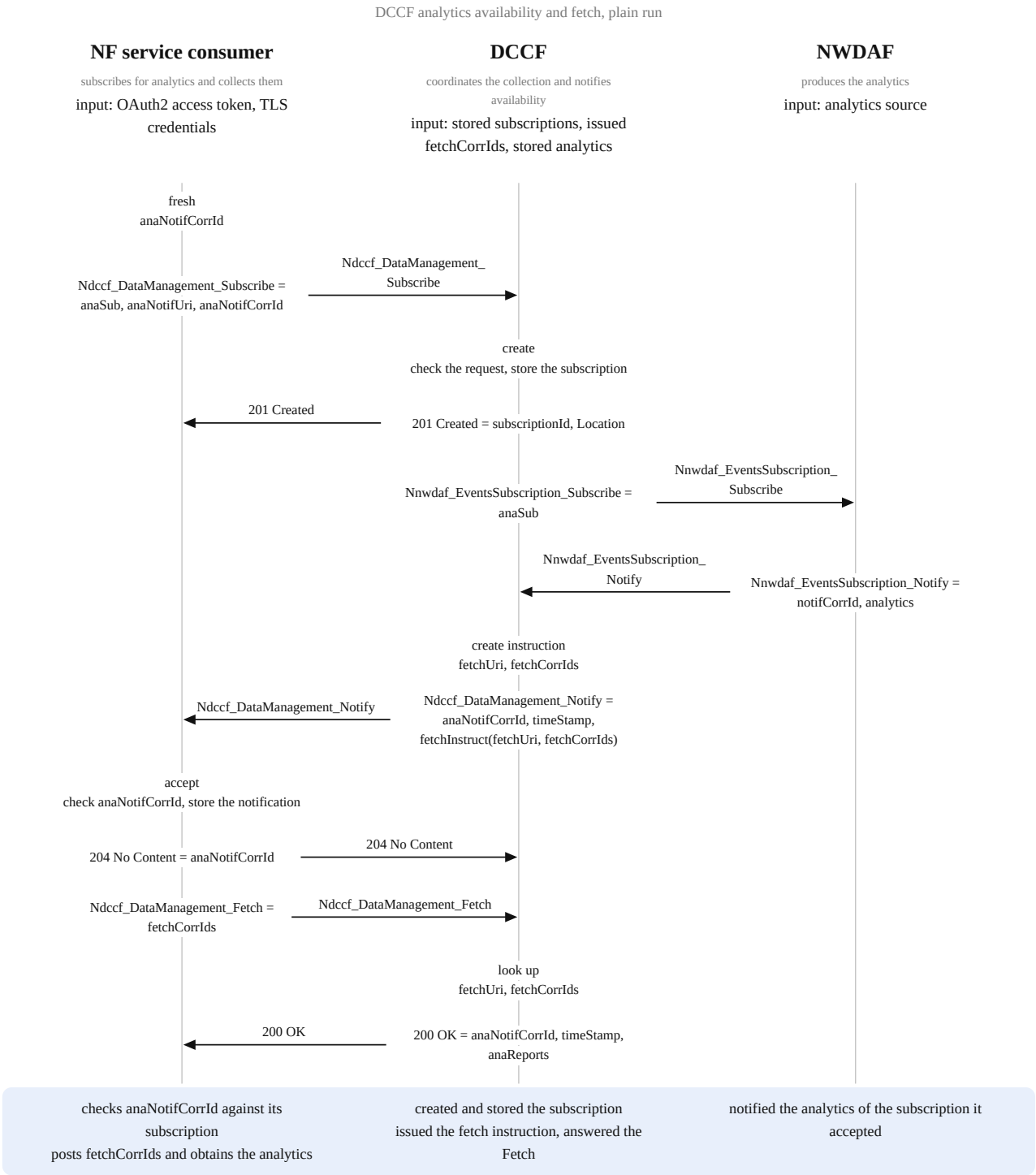


Figure 1. The protocol as specified.

The scope of the analysis is in Appendix D: what was analysed, what was left out, and the sections of the documents it is built from.

2 Findings

The analysis has 1 finding: what follows is not stated in the documents. For 1 of them it is unclear whether the documents state it.

Within the scope analysed the flow gives what it is meant to give: the DCCF acts only for authenticated NF instances, a consumer's notification and Fetch are tied to its own subscription, and no consumer obtains another's analytics. One finding: the DCCF keeps answering a fetch instruction issued before the subscription was updated, because nothing in the text withdraws it. The isolation of consumers rests on the secrecy and unpredictability of the fetch correlation identifiers, which the text types as plain strings and never binds to the requesting NF. All eleven targets have an answer, ten proved on the model of any number of sessions and one shown as an attack on the unrolled model. The proofs rest on TLS and OAuth2 as modelled, and the checks on values that change are finished only for one subscription with one update.

The attack figures use the columns of Figure 1. Red is what the attacker does. Yellow marks a message that a party sends although the specified exchange does not have it at that point, and the outcome that the attack breaks.

2.1 A fetch instruction outlives the subscription state it was issued for

Kind	Weakness of the protocol as specified
New	Unclear: the documents state it in part, or could not be searched
Value	low
Properties	U03 (section 3)
Source text	s0077, s0079, s0080, s0149, s0162
Shown by	ProVerif attack query `u03_old`; Figure 2

What happens. The DCCF answers a Fetch whose fetchUri and fetchCorrIds were issued before the subscription was modified, after the modification replaced the stored instruction with a new one. Clause 4.2.2.5.2 requires the DCCF to find the analytics according to the request and names no error for an instruction that is no longer current. The trace was shown on the unrolled model, and a change that withdraws the identifiers on PUT removes it.

What the attacker needs. The consumer keeps a fetch instruction it received before it modified its subscription and presents it again. No attacker is needed: the DCCF answers it with the stored analytics.

What the attacker gains. Analytics of a subscription state the consumer no longer holds stays retrievable with an instruction the text never withdraws; any party that later obtains the instruction retrieves it, and the consumer is never told its instruction is stale.

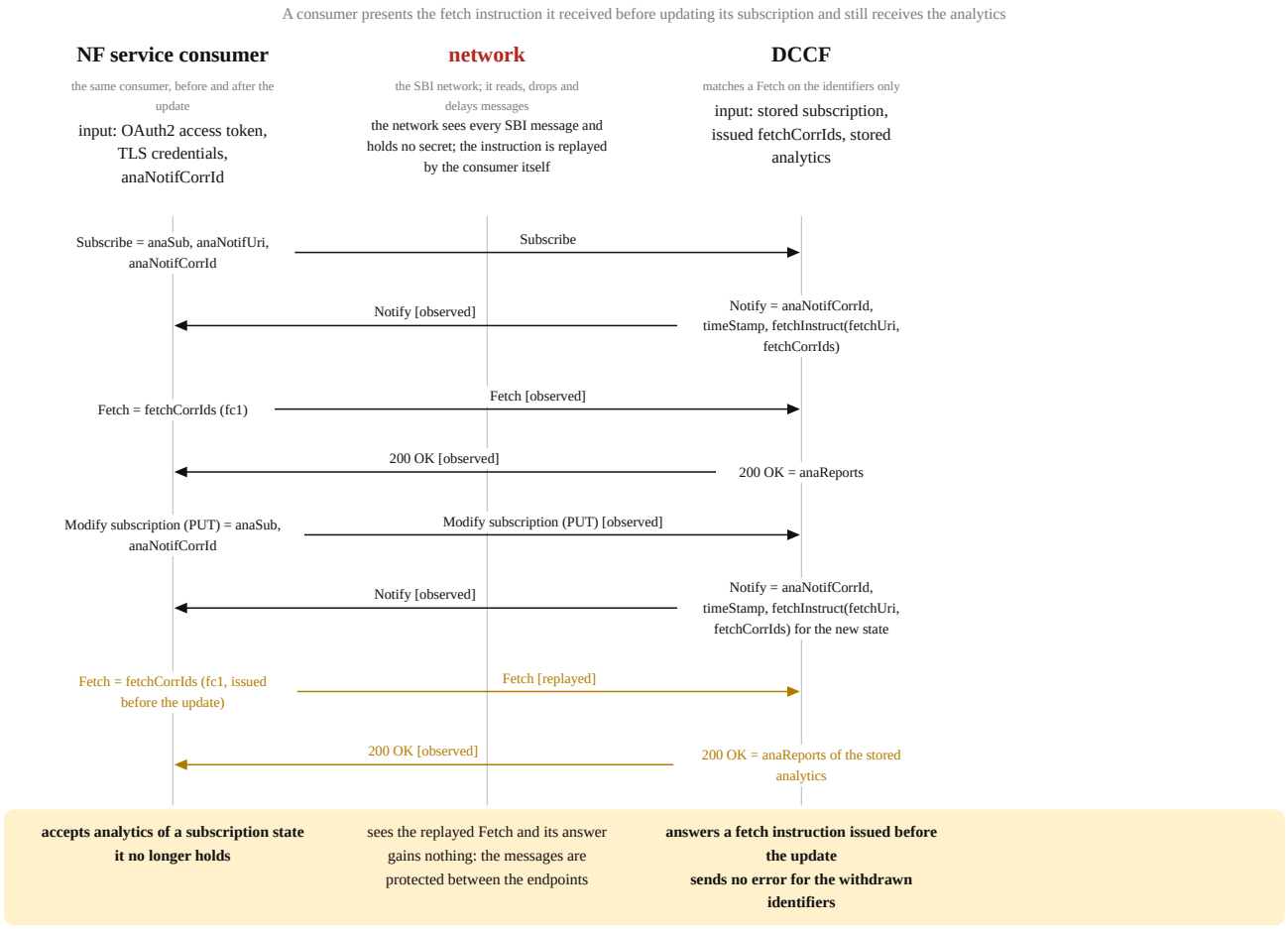


Figure 2. A consumer presents the fetch instruction it received before updating its subscription and still receives the analytics. Property U03; attack query u03_old.

Proposed change. On a subscription modification (PUT) and on deletion (DELETE) the DCCF withdraws the fetch correlation identifiers issued before the change, and a Fetch carrying a withdrawn identifier is answered with an HTTP error response instead of the stored analytics.

Check of the change with ProVerif. The change was put into a copy of the model: On a subscription update the DCCF withdraws the fetch correlation identifiers it issued before the update: the subscription modification (PUT) replaces the stored instruction, and a Fetch that carries the earlier fetchUri or fetchCorrIds is not answered. On the changed model ProVerif proves `u03_old`: the attack is no longer possible there. This is a first check on the model: it shows that the change removes this attack there, not that it introduces no other problem. Changed model: `records/fixes/0053-bounded-fix.pv`.

3 Properties examined

Each row is one property put to ProVerif. The result is ProVerif's answer on the analyst's model of the protocol; value and whether the documents already state the result are the analyst's judgement, with the reasons in Appendix A.

ID	Property	Compromise assumed	Result	Value	In the documents
A02	**peer identity authentication.** Whenever the DCCF accepts a subscription or Fetch request, the claimed consumer NF instance must actually have run the flow towards that DCCF; whenever the consumer accepts a notification, the DCCF must actually have run the flow towards that consumer. Only identity matching is required, not agreement of specific parameters.	no compromise	Holds	low	yes
A03	**correlation identifier agreement.** For every notification the consumer accepts there is a real run of the DCCF whose subscription identifier and anaNotifCorrId agree with the consumer's local values; for every Fetch request the DCCF processes there is a real run of the consumer whose fetchCorrIds agree with the values the DCCF previously gave in a notification.	no compromise	Holds	low	unclear
A04	**no repeated acceptance of a message.** Every notification the consumer accepts and every Fetch request the DCCF processes corresponds to a distinct real run of the peer; no logical message is accepted twice.	no compromise	Holds	low	unclear
A08	**subscription parameter binding.** The subscription parameters the DCCF accepts correspond to the values the consumer really proposed; the anaNotifUri and anaNotifCorrId in the notification agree with those at subscription time; the analytics in the Fetch response correspond to the subscription pointed to by the fetchCorrIds in the request.	no compromise	Holds	low	unclear
K01	**secrecy of access token and TLS keys.** The attacker cannot derive the consumer's OAuth2 token or the TLS session keys between the DCCF and the consumer, and cannot use the token for unauthorised services or operations.	no compromise	Holds	low	yes
Z01	**authorization before protected operations.** Every protected operation the DCCF commits (creating a subscription, delivering a notification, answering a Fetch) must have a valid authorisation issued by the NRF covering that operation and resource; a request without valid authorisation cannot produce these operations.	no compromise	Holds	medium	unclear
Z03	**authorization binding to resource and operation.** For every authorised operation, the authorisation subject, audience, resource (subscription identifier), operation (subscribe/notify/Fetch) and usage context must agree with those at authorisation time; the token or associated identifier cannot be moved to another consumer, another subscription or another operation.	no compromise	Holds	medium	unclear
Z05	**isolation between consumer domains.** Allowing control of one consumer domain (its subscriptions, notifications, Fetch requests), re-check against another consumer domain: that consumer cannot obtain analytics of subscriptions that are not its own, nor use its own fetchCorrIds to retrieve another subscription's data.	no compromise	Holds	medium	unclear
U01	**upstream origin of fetched analytics.** Every Fetch acceptance by consumer C corresponds to the DCCF receiving analytics from the upstream NWDAF named by C and sending C a notification containing the same fetchCorrIds.	no compromise	Holds	low	no
U02	**fetch belongs to the requesting consumer.** For every Fetch acceptance by consumer B, the fetchCorrIds and the subscription belong to B; B cannot retrieve A's analytics.	no compromise	Holds	medium	unclear
U03	**notification and fetch tied to current subscription.** Every notification or Fetch acceptance by consumer C corresponds to C's currently valid subscription, with consistent correlation identifiers; notifications/instructions of an old or already updated subscription cannot be accepted.	no compromise	Fails (1 attack)	low	unclear

The questions asked, and the analyst's answers:

Question	Answer	Rests on
Q1: Upstream analytics, downstream fetchCorrId and binding to the consumer identity	The upstream path is sound: the analytics a consumer fetches were received by the DCCF from the NWDAF on that consumer's notification path, and the instruction it acts on is the one the DCCF issued for it. The binding of the fetch correlation identifiers to the consumer identity is not enforced: the DCCF matches the identifiers only, so the guarantee rests on the identifiers being unguessable and on TLS, although the text types them as strings.	U01, U02, Z03
Q2: Fetch isolation between different legitimate NFs	Consumer B cannot retrieve consumer A's analytics: every Fetch needs a valid token for the requesting NF and the identifiers A was given, which B never receives, and a probe with values the network supplies returns nothing. The isolation is the secrecy of the identifiers, not a check that the requester owns the subscription.	Z05, U02, Z03
Q3: Old notifications/instructions are not associated with a different or updated subscription	A notification cannot be attached to another subscription because the consumer compares the anaNotifCorrId with its own value, but an instruction issued before the subscription was updated is still answered after the update, so the tie to a current subscription is only half there.	U03, A03

4 Limits of this analysis

A result here is a result on a model of the protocol: it says what the specified messages and checks give or fail to give, with cryptography taken as ideal, and nothing about an implementation.

What this analysis does not cover, as the analyst states it:

- TLS is modelled as authenticated encryption under a key derived from the two endpoint identities; TS 29.500 is not part of the modelled text, and TLS replay protection, renegotiation and certificates are not modelled.
- OAuth2 is modelled as an NRF-issued token whose subject must equal the requesting NF; token expiry, audience, the scope string and the client credentials exchange are not modelled.
- The DCCF handles one subscription in one process instance per session; its subscription table, concurrent access and the storage handling of clause 5.1.6.2.15 are not modelled.
- The unrolled model covers one subscription with one update; deletion, repeated updates and several notifications for one subscription are outside it.
- One NWDAF and one consumer pair are modelled: MFAF, multiple upstream reuse and the ADRF are excluded by the scope, and the NWDAF named in targetNfId or targetNfSetId is not varied.
- The user consent branch (Nudm_SDM, 403 USER_CONSENT_NOT_GRANTED) and the 400 SUBSCRIPTION_CANNOT_BE_SERVED path are not modelled.
- Redirect responses (307, 308) are not modelled.
- Truthfulness of the analytics is not modelled: an NWDAF under the attacker's control could supply any value and no target asks for its correctness.
- The strength of the primitives is not modelled, and the correlation identifiers are treated as unguessable names although the text types them as strings.

What the host records:

- The answers of A04, U03 rest on an operation the model simplifies (Appendix D).
- Proved without a control query (Appendix C): A02: a02_notif, A02: a02_fetch, A04: a04_fetch, K01: k01_tok, U03: u03_notif. A control is a variant of the property that must fail; without one, nothing shows that the property as written can fail in this model.

What stays open for each property, and what would settle it, is in Appendix A.

Appendix A: Reading of each property

The analyst's own reading of the prover's answers, not an answer of the prover: what the answers mean for the protocol, what the reading rests on, whether it can be used as it stands, and what it is worth.

A02: peer identity authentication

Whenever the DCCF accepts a subscription or a Fetch request, the claimed consumer NF instance must actually have run that procedure against that DCCF; whenever the consumer accepts a notification, the DCCF must actually have run that procedure against that consumer. Only the identity must match; the specific parameters need not agree.

The DCCF acts only on subscriptions and Fetch requests that come over the TLS session of the NF instance whose identity the message claims, and the consumer accepts a notification only from the DCCF of its subscription. Identity alone is checked; nothing agrees the parameters of the exchange.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value low: TS 33.501 requires mutual authentication between NF service consumer and producer; the proof confirms the message design did not lose that binding.
- Stated in the documents: yes. Where the documents say it: ts-33-501:2871
- Rests on: TLS is modelled as authenticated encryption under a private function of the two endpoint identities; the OAuth2 token is modelled as an NRF value no other party can produce
- Next step: Model the token subject check and the TLS binding from TS 29.500 to confirm that the identity the DCCF uses is the one its peer authenticated with.

A03: correlation identifier agreement

For every notification the consumer accepts, there is a real run of the DCCF whose subscription identifier and anaNotifCorrId match the consumer's local values; for every Fetch request the DCCF handles, there is a real run of the consumer whose fetchCorrIds match the values the DCCF previously gave in a notification.

The notification the consumer accepts carries the notification correlation identifier of a subscription the DCCF really created for it, and a Fetch is served only for a fetch correlation identifier the DCCF really issued in a notification.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value low: the documents state the correlation requirement but never as a security goal; the agreement follows from matching on fresh values.
- Stated in the documents: unclear
- Rests on: the correlation identifiers are modelled as fresh unguessable values; one DCCF instance per subscription; the anaNotifCorrId is echoed in the 201 response, which is how the model binds the answer to the request
- Next step: none for the correlation itself; the currency of an identifier is answered under U03.

A04: no repeated acceptance of a message

Every notification the consumer accepts and every Fetch request the DCCF handles corresponds to a distinct real run of the peer; the same logical message is never accepted twice.

In the modelled run each notification is accepted once and each instruction is served once, so no message is acted on twice. Repetition is excluded by the shape of the model, not by a rule of the text; the re-use of an instruction is shown under U03.

- Result: Holds; answers what is asked: partly; usable with caveats
- Value low: a proof over a session that answers each request once says little about repetition, and the text states no freshness rule.
- Stated in the documents: unclear
- Rests on: TLS replay protection is not modelled; the flow is a linear session per subscription, so a repeated message has to be modelled by unrolling
- Next step: A model of TLS records would settle whether a protected message can be replayed at all; the unrolled model would show whether an instruction may be used more than once on purpose.

A08: subscription parameter binding

The subscription parameters the DCCF accepts correspond to the values the consumer really proposed; the anaNotifUri and anaNotifCorrId in the notification match those at subscription time; the analytics in the Fetch response correspond to the subscription pointed to by the fetchCorrIds in the request.

The notification the consumer accepts carries the notification address and correlation identifier it proposed, and the subscription the DCCF stored is the one that was requested. The data delivered on a Fetch is the stored analytics of that subscription.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value low: binding by fresh values is structural; the answer shows the messages carry the right parameters.
- Stated in the documents: unclear
- Rests on: one DCCF instance per subscription; the subscription contents are taken from the request body as clause 4.2.2.2.2 requires
- Next step: none.

K01: secrecy of access token and TLS keys

The attacker cannot derive the consumer's OAuth2 token or the TLS session keys between the DCCF and the consumer, and cannot use the token for an unauthorized service or operation.

The consumer's access token and the TLS session keys stay out of the attacker's hands, and a token of one NF is not accepted for a request of another NF.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value low: the result follows from the modelled primitives; the documents list the authorization token as a value to be confidentiality protected.
- Stated in the documents: yes. Where the documents say it: ts-33-501:3071
- Rests on: the token is a private function value in the model; TLS is modelled as authenticated encryption
- Next step: none.

Z01: authorization before protected operations

Every protected operation the DCCF submits (creating a subscription, delivering a notification, answering a Fetch) must have a valid authorization issued by the NRF that covers that operation and resource; a request without a valid authorization cannot produce these operations.

Every subscription and every Fetch needs a valid token of the requesting NF. Because the API defines one scope for the whole service, the authorization cannot distinguish a subscribe from a fetch, or one consumer's subscription from another's.

- Result: holds; answers what is asked: partly; usable with caveats
- Value medium: it shows mechanically that the single scope of clause 5.1.9 leaves the operation level unprotected.
- Stated in the documents: unclear
- Rests on: the modelled check is that the token subject equals the requesting identity; token expiry, audience and the scope string are not modelled
- Next step: A scope and audience model from TS 29.501 would settle whether anything at all can be bound per resource.

Z03: authorization binding to resource and operation

For every authorized operation, the authorized subject, audience, resource (subscription identifier), operation (subscribe/notify/Fetch) and context of use must match those at authorization time; the token or associated identifier must not be moved to another consumer, another subscription or another operation.

A token is bound to the NF whose identity it carries, so a consumer cannot act on another consumer's token. The binding of a fetch correlation identifier to a consumer or an operation is not enforced: the DCCF matches the identifiers only, and they are typed as strings, so secrecy is the whole protection.

- Result: holds; answers what is asked: partly; usable with caveats
- Value medium: it exposes that resource binding rests on an unstated assumption, which an editor should decide about.
- Stated in the documents: unclear
- Rests on: the identifiers are modelled as unguessable names although the text types them as strings; one scope for the whole API
- Next step: A model that lets the attacker guess the correlation identifiers would settle what remains if they are predictable.

Z05: isolation between consumer domains

Having control of one consumer domain (its subscriptions, notifications, Fetch requests) gives nothing on another consumer domain: that consumer cannot obtain analytics for subscriptions that are not its own, nor use its own fetchCorrIds to retrieve another subscription's data.

A consumer domain cannot obtain another consumer's analytics in the model, and a legitimate NF probing the fetch endpoint obtains nothing. That holds because the identifiers never leave the consumer they were issued to, not because the DCCF checks who owns the subscription.

- Result: holds; answers what is asked: partly; usable with caveats
- Value medium: the isolation is real in the model but rests on an assumption the text does not state, which a reader of this API needs to know.

- Stated in the documents: unclear
- Rests on: the fetch correlation identifiers are unguessable fresh names; TLS keeps the instruction inside the consumer's session; the DCCF does not compare the requester with the subscriber, as the text does not require it
- Next step: Model the correlation identifiers as guessable to see what a malicious NF gains then.

U01: upstream origin of fetched analytics

Every Fetch acceptance by consumer C corresponds to the DCCF having received analytics from the upstream NWDAF named by C and having sent C a notification containing the same fetchCorrIds.

The analytics a consumer fetches were received by the DCCF from the NWDAF on that consumer's notification path, and the instruction it acts on is the one issued for that notification. With one NWDAF in the model the choice of the upstream NWDAF itself is not tested.

- Result: holds; answers what is asked: partly; usable with caveats
- Value low: searched the documents for a statement that the DCCF must use the NWDAF named in the subscription and found none.
- Stated in the documents: no
- Rests on: a single NWDAF identity; the notification path is a linear session
- Next step: Model a second NWDAF and the targetNfId attribute to ask whether the DCCF must use the NWDAF the consumer named.

U02: fetch belongs to the requesting consumer

Every Fetch acceptance by consumer B has fetchCorrIds and a subscription that belong to B; B cannot retrieve A's analytics.

The analytics a consumer obtains belong to a subscription that consumer created, so one consumer cannot retrieve another consumer's analytics in the modelled flow. The DCCF never compares the requester with the subscriber; the answer comes from the identifiers never being shared.

- Result: holds; answers what is asked: partly; usable with caveats
- Value medium: it states plainly that ownership of a subscription is not a checked property of the Fetch operation.
- Stated in the documents: unclear
- Rests on: the identifiers are unguessable; TLS keeps the instruction in the consumer's session
- Next step: Model the identifiers as guessable, or a consumer that reads another consumer's notification, to see whether the missing owner check can be used.

U03: notification and fetch tied to current subscription

Every notification or Fetch acceptance by consumer C corresponds to a currently valid subscription of C, and the associated identifiers agree; old notifications/instructions from an old or updated subscription cannot be accepted.

The notification part holds: a consumer acts only on a notification of a subscription it created and compares its own correlation identifier. The currency part fails: after the subscription is updated the DCCF still answers the instruction issued before the update, and the consumer is not told it is stale.

- Result: fails (1 attack); answers what is asked: partly; usable with caveats
- Value low: the gap is real but modest: it needs a consumer that keeps an old instruction, and nothing in the text forbids returning it.

- Stated in the documents: unclear
- Rests on: the unrolled model runs one update, not a deletion; the source states no invalidation rule; the stage-2 specification describes a deadline in the fetch instruction that this text does not carry
- Next step: Model the deletion of a subscription and a second update to see whether the same holds for DELETE and repeated updates.

Appendix C: Claims and prover answers

The prover's record for each target: the level reached, the query that reached it, whether a control was checked, and the attacks with the analyst's verdict on each.

target	level	bound	lemma	control	attacks	claims
A02	proved		a02_sub	3 of 5		5 proved
A03	proved		a03_notif	checked		3 proved
A04	proved		a04_notif	3 of 4		4 proved
A08	proved		a08_notif	checked		2 proved
K01	proved		k01_tok	2 of 4		4 proved
Z01	proved		z01_sub	checked		3 proved
Z03	proved		z03_tok	checked		3 proved
Z05	proved		z05_data	checked		2 proved
U01	proved		u01_path	checked		3 proved
U02	proved		u02_data	checked		1 proved
U03	proved		u03_notif	1 of 2	u03_old (depends_on_ambiguity)	2 proved, 1 attack

Claim by claim, in the analyst's wording of each claim; the answer to each is the prover's:

target	claim	attacker	answer	query	control
A02	Each subscription the DCCF accepts was proposed by the very NF instance that ran the Subscribe flow towards it (identity match only, no parameter agreement); the attacker has the network only.	`network`	proved	a02_sub	checked
A02	Each notification the consumer accepts was sent by the DCCF to that consumer for that correlation identifier, so the consumer's peer is the DCCF of its subscription; the attacker has the network only.	`network`	proved	a02_notif	none
A02	Each Fetch the DCCF serves was requested by the consumer whose own Fetch request it is, so a legitimate NF with its own credentials cannot make the DCCF serve it a request it did not make.	`malicious_nf`	proved	a02_fetch	none
A02	Each notification the consumer accepts was sent by the DCCF to that consumer for that correlation identifier; the attacker has the network only.	`network`	proved	a02_notif	checked
A02	Each Fetch the DCCF serves was requested by the consumer whose own Fetch request it is; the attacker is a legitimate NF with its own credentials.	`malicious_nf`	proved	a02_fetch	checked
A03	Each accepted notification carries the anaNotifCorrId of a subscription the DCCF really created for that consumer.	`network`	proved	a03_notif	checked
A03	Each Fetch the DCCF serves carries a fetch correlation identifier the DCCF really issued earlier in a notification.	`network`	proved	a03_fetch	checked

target	claim	attacker	answer	query	control
A03	A legitimate NF that presents fetch correlation identifiers cannot have the DCCF serve a Fetch for an identifier it was not issued.	`malicious_nf`	proved	a03_fetch	checked
A04	A Fetch is served at most once per issued fetch correlation identifier, also when a legitimate NF offers identifiers it was given; the attacker has its own credentials only.	`malicious_nf`	proved	a04_fetch	none
A04	A notification is accepted by the consumer at most once per notification the DCCF issued.	`network`	proved	a04_notif	checked
A04	A Fetch is served at most once per fetch correlation identifier the DCCF issued.	`network`	proved	a04_fetch	checked
A04	A Fetch is served at most once per issued fetch correlation identifier, also when a legitimate NF offers identifiers it was given.	`malicious_nf`	proved	a04_fetch	checked
A08	The notification the consumer accepts carries the anaNotifUri and anaNotifCorrId values the consumer itself proposed in its subscription.	`network`	proved	a08_notif	checked
A08	A legitimate NF cannot make the DCCF deliver a notification carrying another consumer's correlation identifier to its own endpoint.	`malicious_nf`	proved	a08_notif	checked
K01	The NF service consumer's OAuth2 access token stays secret from an attacker on the network.	`network`	proved	k01_tok	none
K01	A legitimate NF with its own credentials does not learn another consumer's access token.	`malicious_nf`	proved	k01_tok	none
K01	The TLS session key between a consumer and the DCCF is not derivable by an attacker on the network.	`network`	proved	k01_key	checked
K01	The consumer's OAuth2 access token stays secret from an attacker on the network.	`network`	proved	k01_tok	checked
Z01	A legitimate NF can only have the DCCF act on its own token: the token subject must equal the requesting NF identity.	`malicious_nf`	proved	z03_tok	checked
Z01	The DCCF creates a subscription only after a valid NRF-issued authorization for the requesting identity.	`network`	proved	z01_sub	checked
Z01	The DCCF serves a Fetch only after a valid NRF-issued authorization for the requesting identity.	`network`	proved	z01_fetch	checked
Z03	A token of one consumer is not accepted for a request of another consumer.	`network`	proved	z03_tok	checked
Z03	A fetch correlation identifier is served only to the consumer it was issued to.	`network`	proved	z03_corr	checked
Z03	A legitimate NF cannot have the DCCF serve it a fetch correlation identifier that was issued to another consumer.	`malicious_nf`	proved	z03_corr	checked
Z05	The DCCF sends notified analytics only to a consumer it issued a fetch instruction to, so one consumer domain does not obtain another consumer's analytics.	`malicious_nf`	proved	z05_data	checked
Z05	A legitimate NF that probes the Fetch endpoint with values the network supplies never obtains any fetched analytics.	`malicious_nf`	proved	z05_never	checked

target	claim	attacker	answer	query	control
U01	A legitimate NF cannot obtain analytics that came from an NWDAF on another consumer's notification path.	`malicious_nf`	proved	u01_path	checked
U01	The analytics a consumer obtains were received by the DCCF from the upstream NWDAF on that consumer's notification path.	`network`	proved	u01_path	checked
U01	The analytics a consumer fetches are the ones the NWDAF of its subscription produced, not a value made up on the network.	`network`	proved	u01_data	checked
U02	The analytics a consumer obtains with a Fetch belong to a subscription that consumer created, so one consumer cannot retrieve another consumer's analytics.	`malicious_nf`	proved	u02_data	checked
U03	A notification the consumer acts on belongs to a subscription the DCCF created for it.	`network`	proved	u03_notif	none
U03	A fetch instruction the consumer acts on belongs to a subscription that is current at that moment: an instruction issued before the subscription was updated is not accepted.	`malicious_nf`	attack (depends_on_ambiguity)	u03_old	
U03	A notification a consumer acts on was sent to it by the DCCF for that subscription, not supplied by another NF.	`malicious_nf`	proved	u03_notif	checked

Appendix D: How the analysis was run

An LLM analyst builds the model of the protocol from the documents and puts each property to ProVerif; every result in section 3 is an answer of the prover.

ProVerif+ Agent session session-20261115T094233, 2026-11-15. Status: finished. Model: deepseek-v4-1-flash-261017. Steps: 69, prover 5 s, wall 13 min, cost 0.112 USD.

ProVerif: 118 proofs run, 118 decided, 0 without an answer, 4 s, 53 answers recorded.

What was analysed

Protocol	DCCF analytics availability and fetch
Version	19.7.0
Analysed	NF consumer, DCCF, NWDAF; fixed non-UE aggregated analytics, one subscription, upstream notification, downstream availability/fetch instructions, Fetch and data response; notify correlation and fetch correlation ID are kept, MFAF and reuse of multiple upstreams are excluded.
Not analysed	The symbolic model does not prove the computational security or post-quantum resistance of the algorithms; attack traces hold only under explicit assumptions and are not directly equivalent to specification vulnerabilities. Every property must have a reachability/positive control. Fixed to Rel-19. The external network attacker and the malicious legitimate NF holding its own credentials are distinguished separately; TLS authenticated encryption is modelled per endpoint, and a legitimate NF cannot read another NF's TLS session. OAuth2 subject/audience/scope/expiry and TLS identity binding follow the attached baseline; a service scope is not automatically equal to resource ownership. Monotonic tables cannot model real deletion, and finite-state lifecycles must be explicitly bounded. The target is the specified API sub-flow, not full 5G security.
Source text	1271 numbered passages (928151 bytes) from the sections listed below. A reference such as s0001 in this report is one of these passages

The sections of the documents the analysis is built from:

Document	Section	Title
main	4.2.2.2.2	Subscription to analytics notifications
main	4.2.2.4.2	Notification of subscribed analytics
main	4.2.2.5	Ndccf_DataManagement_Fetch service operation
main	4.2.2.5.1	General
main	4.2.2.5.2	Retrieving notified analytics and data
main	5.1	Ndccf_DataManagement Service API
main	5.1.1	Introduction
main	5.1.2	Use of HTTP
main	5.1.2.1	General
main	5.1.2.2	HTTP standard headers
main	5.1.2.2.1	General
main	5.1.2.2.2	Content type
main	5.1.2.3	HTTP custom headers

Document	Section	Title
main	5.1.3	Resources
main	5.1.3.1	Overview
main	5.1.3.2	Resource: DCCF Analytics Subscriptions
main	5.1.3.2.1	Description
main	5.1.3.2.2	Resource definition
main	5.1.3.2.3	Resource standard methods
main	5.1.3.2.3.1	POST
main	5.1.3.2.4	Resource custom operations
main	5.1.3.3	Resource: Individual DCCF Analytics Subscription
main	5.1.3.3.1	Description
main	5.1.3.3.2	Resource definition
main	5.1.3.3.3	Resource standard methods
main	5.1.3.3.3.1	PUT
main	5.1.3.3.3.2	DELETE
main	5.1.3.3.4	Resource custom operations
main	5.1.3.4	Resource: DCCF Data Subscriptions
main	5.1.3.4.1	Description
main	5.1.3.4.2	Resource definition
main	5.1.3.4.3	Resource standard methods
main	5.1.3.4.3.1	POST
main	5.1.3.4.4	Resource custom operations
main	5.1.3.5	Resource: Individual DCCF Data Subscription
main	5.1.3.5.1	Description
main	5.1.3.5.2	Resource definition
main	5.1.3.5.3	Resource standard methods
main	5.1.3.5.3.1	PUT
main	5.1.3.5.3.2	DELETE
main	5.1.3.5.4	Resource custom operations
main	5.1.3.6	Void
main	5.1.3.7	Void
main	5.1.4	Custom operations without associated resources
main	5.1.4.1	Overview
main	5.1.4.2	Operation: transfer-data-sub
main	5.1.4.2.1	Description
main	5.1.4.2.2	Operation definition
main	5.1.5	Notifications

Document	Section	Title
main	5.1.5.1	General
main	5.1.5.2	Analytics notification
main	5.1.5.2.1	Description
main	5.1.5.2.2	Target URI
main	5.1.5.2.3	Standard methods
main	5.1.5.2.3.1	POST
main	5.1.5.3	Data notification
main	5.1.5.3.1	Description
main	5.1.5.3.2	Target URI
main	5.1.5.3.3	Standard methods
main	5.1.5.3.3.1	POST
main	5.1.5.4	Fetch notification
main	5.1.5.4.1	Description
main	5.1.5.4.2	Target URI
main	5.1.5.4.3	Standard methods
main	5.1.5.4.3.1	POST
main	5.1.6	Data model
main	5.1.6.1	General
main	5.1.6.2	Structured data types
main	5.1.6.2.1	Introduction
main	5.1.6.2.2	Type NdccfAnalyticsSubscription
main	5.1.6.2.3	Type NdccfDataSubscription
main	5.1.6.2.4	Type NdccfAnalyticsSubscriptionNotification
main	5.1.6.2.5	Type NdccfDataSubscriptionNotification
main	5.1.6.2.6	Type FormattingInstruction
main	5.1.6.2.7	Type ProcessingInstruction
main	5.1.6.2.8	Type ParameterProcessingInstruction
main	5.1.6.2.9	Type NotifSummaryReport
main	5.1.6.2.10	Type EventParamReport
main	5.1.6.2.11	Type ReportingOptions
main	5.1.6.2.12	Void
main	5.1.6.2.13	Type DccfEvent
main	5.1.6.2.14	Type NotifyEndpoint
main	5.1.6.2.15	Type: StorageHandlingInformation
main	5.1.6.2.16	Type: DeletionAlert
main	5.1.6.2.17	Type: NotifResponse

Document	Section	Title
main	5.1.6.2.18	Void
main	5.1.6.2.19	Type: DataTransferResp
main	5.1.6.3	Simple data types and enumerations
main	5.1.6.3.1	Introduction
main	5.1.6.3.2	Simple data types
main	5.1.6.3.3	Enumeration: SummarizationAttribute
main	5.1.6.3.4	Enumeration: AggregationLevel
main	5.1.6.3.5	Enumeration: DataCollectionPurpose
main	5.1.6.3.6	Enumeration: TermCause
main	5.1.6.4	Data types describing alternative data types or combinations of data types
main	5.1.6.5	Binary data
main	5.1.7	Error handling
main	5.1.7.1	General
main	5.1.7.2	Protocol errors
main	5.1.7.3	Application errors
main	5.1.8	Feature negotiation
main	5.1.9	Security
ts-29-520	4.2.1	Service description
ts-29-520	4.2.1.1	Overview
ts-29-520	4.2.1.2	Service architecture
ts-29-520	4.2.1.3	Network functions
ts-29-520	4.2.1.3.1	Network Data Analytics Function (NWDAF)
ts-29-520	4.2.1.3.2	NF service consumers
ts-29-520	4.2.2.2	Nnwdaf_EventsSubscription_Subscribe service operation
ts-29-520	4.2.2.2.1	General
ts-29-520	4.2.2.2.2	Subscription to event notifications
ts-29-520	4.2.2.2.3	Updating a subscription to event notifications
ts-29-520	4.2.2.3	Nnwdaf_EventsSubscription_Unsubscribe service operation
ts-29-520	4.2.2.3.1	General
ts-29-520	4.2.2.3.2	Unsubscribing from event notifications
ts-29-520	4.2.2.4	Nnwdaf_EventsSubscription_Notify service operation
ts-29-520	4.2.2.4.1	General
ts-29-520	4.2.2.4.2	Notification of a subscribed event
ts-29-520	5.1.1	Introduction
ts-29-520	5.1.2	Use of HTTP
ts-29-520	5.1.2.1	General

Document	Section	Title
ts-29-520	5.1.2.2	HTTP standard headers
ts-29-520	5.1.2.2.1	General
ts-29-520	5.1.2.2.2	Content type
ts-29-520	5.1.2.3	HTTP custom headers
ts-29-520	5.1.3.1	Resource structure
ts-29-520	5.1.3.2	Resource: NWDAF Events Subscriptions
ts-29-520	5.1.3.2.1	Description
ts-29-520	5.1.3.2.2	Resource definition
ts-29-520	5.1.3.2.3	Resource standard methods
ts-29-520	5.1.3.2.3.1	POST
ts-29-520	5.1.3.2.4	Resource custom operations
ts-29-520	5.1.3.3	Resource: Individual NWDAF Event Subscription
ts-29-520	5.1.3.3.1	Description
ts-29-520	5.1.3.3.2	Resource definition
ts-29-520	5.1.3.3.3	Resource standard methods
ts-29-520	5.1.3.3.3.1	DELETE
ts-29-520	5.1.3.3.3.2	PUT
ts-29-520	5.1.3.3.4	Resource custom operations
ts-29-520	5.1.5	Notifications
ts-29-520	5.1.5.1	General
ts-29-520	5.1.5.2	Event notification
ts-29-520	5.1.5.2.1	Description
ts-29-520	5.1.5.2.2	Operation definition
ts-29-520	5.1.6.2.2	Type NnwdafEventsSubscription
ts-29-520	5.1.6.2.3	Type EventSubscription
ts-29-520	5.1.6.2.4	Type NnwdafEventsSubscriptionNotification
ts-29-520	5.1.6.2.5	Type EventNotification
ts-29-520	5.1.6.2.7	Type EventReportingRequirement
ts-29-520	5.1.6.2.22	Type NetworkPerfRequirement
ts-29-520	5.1.6.2.23	Type NetworkPerfInfo
ts-29-520	5.1.6.2.30	Type ThresholdLevel
ts-29-520	5.1.6.2.49	Type ConsumerNfInformation
ts-29-520	5.1.6.3.3	Enumeration: NotificationMethod
ts-29-520	5.1.6.3.4	Enumeration: NwdafEvent
ts-29-520	5.1.6.3.10	Enumeration: NetworkPerfType
ts-29-520	5.1.6.3.26	Enumeration: TermCause

Document	Section	Title
ts-29-520	5.1.7	Error handling
ts-29-520	5.1.7.1	General
ts-29-520	5.1.7.2	Protocol errors
ts-29-520	5.1.7.3	Application errors
ts-29-520	5.1.8	Feature negotiation
ts-29-520	5.1.9	Security

As the source step established, the purpose of the protocol is as follows: NF consumer, DCCF, NWDAF; fixed non-UE aggregated analytics, one subscription, upstream notifications, downstream availability/fetch instructions, Fetch and data response; notify correlation and fetch correlation ID are retained; MFAF and multiplexing across multiple upstreams are excluded.

The properties against the documents

Before writing a query, the analyst checked the drafted list of properties against what the documents themselves state the protocol is for. The name, the subject of each property, and what the documents say reflect the analyst's reading; the locations named do exist in the documents.

ID	Name	About	What the documents say	Where
A02	peer identity authentication	the DCCF is assured that the consumer it accepts a Subscribe or Fetch from is the NF instance that really ran the flow towards it, and the consumer is assured the notification comes from the DCCF of its subscription	4.2.2.2.2 requires the subscription information; the binding comes from TS 29.500 TLS and TS 33.501 NF mutual authentication as in the task scope	s0005, s0021, s0050, s0076, s0417
A03	correlation identifier agreement	each accepted notification carries the anaNotifCorrId of a subscription the DCCF really created, and each processed Fetch carries fetchCorrIds the DCCF really issued	s0051, s0337 fix anaNotifCorrId; s0077, s0310 require the fetch correlation identifiers of the notification	s0051, s0337, s0077, s0310
A04	no repeated acceptance of a message	each notification the consumer accepts and each Fetch the DCCF serves corresponds to a separate peer run; nothing is acted on twice	no replay protection is stated; timeStamp marks preparation time only (s0052, s0337)	s0050, s0052, s0077
A08	subscription parameter binding	the DCCF acts on the values the consumer proposed (anaSub, anaNotifUri, anaNotifCorrId) and a Fetch returns the analytics of the subscription its fetchCorrIds point to	s0021, s0027, s0029 on using and storing the request; s0051 on the notification; s0310, s0080 on the fetch response	s0005, s0021, s0051, s0080
K01	secrecy of access token and TLS keys	the consumer's OAuth2 token and the TLS session keys stay secret, and one NF cannot use another's token for an operation it was not issued for	s0417, s0418: client credentials with the NRF as authorization server	s0417, s0418
Z01	authorization before protected operations	each protected DCCF operation runs only for a requester holding a valid NRF-issued authorization covering it	s0417, s0418 make access subject to OAuth2; the text does not say which check the DCCF makes	s0417, s0418
Z03	authorization binding to resource and operation	a token or a fetch correlation identifier stays with its subject, resource and operation and is not used for another consumer, subscription or operation	s0420 defines one scope for the whole service, so resource binding must come from message identifiers (s0051, s0077)	s0420, s0051, s0077
Z05	isolation between consumer domains	a consumer controlling its own subscription and Fetch requests does not obtain another consumer's analytics	s0310, s0080 tie a response to the submitted identifiers; no isolation goal is stated, it comes from the task scope	s0077, s0080

ID	Name	About	What the documents say	Where
U01	upstream origin of fetched analytics	fetched analytics came from the upstream NWDAF of that consumer's subscription and the instruction carries that notification's identifiers	s0014 lets the consumer name the target NWDAF and s0021, s0026 let the DCCF subscribe to it; no origin goal is stated	s0014, s0021, s0026, s0077
U02	fetch belongs to the requesting consumer	the analytics a consumer obtains belong to a subscription of that consumer	s0310, s0080 say nothing of the requester identity, so the goal is read from the task scope	s0077, s0080
U03	notification and fetch tied to current subscription	a notification or fetch instruction the consumer acts on belongs to a subscription current at that moment, and is not accepted twice	PUT and DELETE exist (s0149, s0162) and fetchCorrIds are only said to be those previously provided (s0077); no invalidation rule is stated	s0077, s0149, s0162

The analyst added no property: the drafted list covers what the documents state.

The security statements of the documents, paragraph by paragraph

The host listed the 398 paragraphs of the sections in which the documents say what the protocol protects and against what, and the analyst said of each what it made of it: a property, an attacker, or why neither. Taken up: 32. Not taken: 366. Which paragraphs are listed follows from the layout of the standard; where each went is the analyst's judgement.

Paragraphs	Where	Taken up as	Why not
P001 to P002	rfc6749:2778-2786	No goal stated	intro
P003 to P006	rfc6749:2790-2823	K01, A02, `network`	
P007 to P010	rfc6749:2827-2850	A02, Z01, `malicious_nf`	
P011 to P015	rfc6749:2859-2881	K01, Z01, Z03, `network`	
P016 to P033	rfc6749:2885-3003	Outside the scope	refresh tokens, authorization codes, redirect URIs and ROPC belong to other grants
P034 to P035	rfc6749:3013-3019	K01, `network`	
P036	rfc6749:3023-3028	A02, `network`, `malicious_nf`	
P037 to P039	rfc6749:3032-3041	K01, `network`	
P040 to P050	rfc6749:3045-3138	Outside the scope	phishing, CSRF and clickjacking need an end user or browser; the modelled NFs have none
P051 to P052	rfc6749:3142-3150	No goal stated	advice to validate input; not askable
P053 to P062	rfc6749:3154-3210	Outside the scope	authorization-server redirect endpoints and the implicit grant
P063	ts-33-501:2491-2491	Outside the scope	UE/gNB bidding down
P064 to P213	ts-33-501:2495-2853	Outside the scope	UE authentication, keys, RAN, UDM, AUSF, UDR and subscriber privacy are outside this API sub-flow

Paragraphs	Where	Taken up as	Why not
P214	ts-33-501:2859-2859	No goal stated	trust-zone assumption
P215 to P220	ts-33-501:2865-2875	A02, Z01, A04, `network`, `malicious_nf`	
P221 to P227	ts-33-501:2879-2891	Z01, K01, `malicious_nf`	
P228 to P241	ts-33-501:2895-2923	Outside the scope	NEF and SCP requirements; neither is a party of the modelled flow
P242 to P312	ts-33-501:2929-3075	Outside the scope	interconnect, SEPP, N32-f and roaming intermediary requirements; no inter-PLMN interface here
P313 to P314	ts-33-501:3079-3081	Outside the scope	IPUPS packet filtering is a transport function
P315 to P318	ts-33-501:3085-3091	No goal stated	originating network header is deployment information
P319 to P337	ts-33-501:3097-3141	No goal stated	operator requirements on monitoring configuration and security visibility/configurability
P338 to P369	ts-33-501:3149-3215	The prover cannot be asked	algorithm identifiers and negotiation: the strength of a primitive is not askable
P370 to P398	ts-33-501:3219-11901	Outside the scope	5G-RG, NSSAAF, N3, EAP-TLS, IAB, MBSF, MBSTF and NEF annex requirements

The parties and the protected values, one by one

The host listed the 29 parties and protected values that the drafted properties name, and the analyst said of each which attacker controls the party or holds the value, or why no such attacker was set. With an attacker: 3. Without: 26. The list follows from the properties; what was made of each entry is the analyst's judgement.

ID	Party or value	Kind	Attacker	Why none
S01	NF service consumer	party	`malicious_nf`	
S02	DCCF	party	Everything lost by definition	a controlled DCCF produces every notification and fetch answer, so no target is asked under it
S03	NWDAF	party	Outside the scope	truthfulness of analytics content is outside this API sub-flow
S04	NRF	party	Everything lost by definition	a compromised authorization server voids all authorization
S05	NF service consumer A	party	`malicious_nf`	
S06	NF service consumer B	party	`malicious_nf`	
S07	subscription resource	value	Everything lost by definition	reading it yields the correlation identifiers, whose secrecy the fetch properties rest on
S08	notification target address anaNotifUri and notification correlation identifier anaNotifCorrId	value	Everything lost by definition	same: the notification identifiers are what ties a notification to a subscription
S09	Fetch correlation identifiers fetchCorrIds	value	Everything lost by definition	the fetch correlation identifiers are the bearer credential of the Fetch
S10	subscriptionId	value	Nothing gained	the subscription identifier is not used as an authenticator

ID	Party or value	Kind	Attacker	Why none
S11	anaNotifCorrId	value	Nothing gained	a correlator, not a credential: producing a notification still needs the DCCF TLS session
S12	fetchCorrIds	value	Everything lost by definition	whoever holds them may fetch, as s0077 and s0080 require
S13	fetchUri	value	Nothing gained	the URI alone is not enough; the body must carry the correlation identifiers
S14	notification message	value	Everything lost by definition	a holder of the notification holds its fetch correlation identifiers
S15	Fetch request	value	Everything lost by definition	a holder of the request holds its fetch correlation identifiers
S16	anaSub	value	Nothing gained	subscription content is configuration, not an authenticator
S17	anaNotifUri	value	Nothing gained	the callback URI is an address, not a credential
S18	targetNfId/targetNfSetId	value	Nothing gained	NF identifiers are discoverable via the NRF
S19	OAuth2 access token	value	Everything lost by definition	a holder of the token is authorized as that consumer for the single scope; K01 already asks that it stay secret
S20	TLS session keys	value	Everything lost by definition	with the session keys the attacker reads and writes that session
S21	scope ndccf-datamanagement	value	Nothing gained	a scope name grants nothing without a token
S22	scope and audience in the OAuth2 token	value	Nothing gained	token metadata grants nothing by itself
S23	analytics in the Fetch response	value	Everything lost by definition	the analytics is the protected value itself
S24	OAuth2 token	value	Everything lost by definition	same as S19
S25	each consumer's subscription resource	value	Everything lost by definition	same as S07
S26	notification content	value	Everything lost by definition	same as S14
S27	upstream analytics	value	Everything lost by definition	the upstream analytics is the value whose delivery is analysed
S28	analytics	value	Everything lost by definition	same as S23
S29	subscription update state	value	Nothing gained	knowing whether a subscription is current gives no credential

The properties under each attacker

The attackers are the analyst's, set before a query was written; each answer was recorded under one of them. A cell is the prover's answer for the property under that attacker; "No answer" is an attacker set for the property under which the prover left a claim undecided or nothing was recorded; an empty cell is an attacker the analyst did not set for the property.

Attacker	Has	Basis	Where
`network`	the network only: it reads, drops, changes and sends messages, and holds no secret of a party		
`malicious_nf`	a legitimate NF service consumer with its own identity and its own NRF-issued token for scope ndccf-datamanagement, its own TLS sessions, and full control of its subscriptions, notification endpoint and Fetch requests	5.1.9 lets any NF with a token consume the API and defines one scope for the whole service (s0417, s0420)	s0417, s0418, s0420

ID	`network`	`malicious_nf`
A02	Holds	Holds
A03	Holds	Holds
A04	Holds	Holds
A08	Holds	Holds
K01	Holds	Holds
Z01	Holds	Holds
Z03	Holds	Holds
Z05		Holds
U01	Holds	Holds
U02		Holds
U03	Holds	Fails

Source and targets

Source: 1271 units, 928151 bytes, frozen in source-delivery-01.json. Targets: 11 approved (A02, A03, A04, A08, K01, Z01, Z03, Z05, U01, U02, U03); the questions asked: 3. The list of targets was approved as drafted, beforehand, with --approve-targets.

Decisions the host took by its documented default (task `decisions: auto`), for you to check:

- approve_package: approved source package protocol-10e4fd6d8a752af4dad43e70f59f

This step as it reports itself: 1 requests (1 failed), 0.000 USD, 388996 tokens in and 5362 out, 277 s.

Notes on the prover and the model

- A04: answered in ProVerif, and the analyst names changing_state for this target
- U03: answered in ProVerif, and the analyst names changing_state for this target

A ProVerif model that simplifies algebra (xor, dh_algebra, other_algebra) gives the attacker less than the real operation: a proof there is a proof against a weaker attacker, an attack there is an attack. A ProVerif model that leaves out a check on a value that changes (changing_state) lets parties accept more: a proof there holds, an attack there may be nothing but the missing check. Tamarin+ Agent models these as they are.

State review by the analyst

This part is the analyst's own reading of the source, not an answer of the prover. "yes" under shown by ProVerif means that ProVerif found a trace with these steps in the unrolled model, so the steps are possible; that they break a target is said only by an attack in the table of answers, and what else they cost is the analyst's reading. "nothing found" says which patterns were looked at and proves nothing.

value	kept by	changes when	checked when	refs
anaNotifCorrId (notification correlation identifier)	the DCCF in the stored subscription and the NF service consumer that proposed it (s0008, s0030)	a subscription is created (POST, s0030) and when it is modified by PUT (s0149, s0155)	the DCCF copies it into each notification (s0051, s0337) and the consumer compares it with its own value when it accepts the notification	s0008, s0051, s0149, s0337
fetchUri and fetchCorrIds (the fetch instruction)	the DCCF with the stored analytics, and the consumer in the notification it stores (s0061, s0077, s0310)	a new notification is issued, and after a subscription update the DCCF issues a new instruction for the same subscription	on every Fetch the DCCF compares the submitted values with the ones it issued (s0077, s0079)	s0076, s0077, s0079, s0080, s0310
the validity of the stored subscription (whether it exists and which contents are current)	the DCCF (s0029)	created by POST (s0026..s0029), modified by PUT (s0149), removed by DELETE (s0162)	the DCCF looks the subscription up when it creates, modifies or deletes it; it is not compared with the instructions already issued when they are used	s0029, s0149, s0162, s0155

pattern	result	target	shown by ProVerif	steps
replay	candidate	U03	yes: ProVerif falsifies `u03_old` on bounded.pv	1. NF service consumer subscribes; DCCF creates the subscription and stores it (s0026..s0029). / 2. DCCF notifies with fetchInstruct (fetchUri, fetchCorrIds) (s0056, s0076, s0077); the consumer fetches and is answered (s0078..s0080). / 3. The consumer modifies the subscription with PUT (s0149..s0161); the DCCF replaces the stored values and issues a new instruction (s0155, s0030). / 4. The consumer presents the earlier fetchUri and fetchCorrIds again; the DCCF finds the stored analytics according to the request (s0079, s0080) and answers 200 OK.
desynchronisation	candidate		yes: ProVerif falsifies `u03_old` on bounded.pv	1. Subscription with instruction fc1 is current on both sides. / 2. PUT replaces the subscription contents; the DCCF issues fc2 (s0149, s0155). / 3. Nothing in the text tells the consumer that fc1 is withdrawn, and the DCCF keeps answering fc1 because it only matches the identifiers of the request (s0077, s0079).
no_rollback_on_failure	candidate		no: not tried with ProVerif	1. The consumer obtains analytics with a fetch instruction. / 2. The subscription is deleted with DELETE (s0162) or abandoned. / 3. The stored analytics and the correlation identifiers it was issued under are not described as withdrawn by the text, so the instruction still names retrievable data.
repeat_one_time_action	nothing found			No step in the flow is meant to happen once: the notification and the Fetch may legitimately repeat for a subscription.
update_before_check	nothing found			The DCCF validates the token of the PUT before replacing the stored subscription values; nothing in the text reverses that order.
race_between_sessions	nothing found			Each subscription has its own correlation identifiers and the fetch is matched against the values of the notification instance, so two runs do not read and change one value.

pattern	result	target	shown by ProVerif	steps
distinguishable_failure	nothing found			The error answers of clause 5.1.7 distinguish reasons (SUBSCRIPTION_CANNOT_BE_SERVED s0021, USER_CONSENT_NOT_GRANTED s0023), but they are returned to the requester about its own request, and the modelled flow has no refusal answer that tells a third party what another consumer keeps.
wrap_or_window	nothing found			The correlation identifiers are strings, not counters, and the flow has no acceptance window: timePeriod concerns the analytics target period, not the acceptance of an instruction.

What the analyst named for targets:

step	targets	features	reason
46	U03, A04	changing_state	U03 and A04 rest on the per-subscription values the DCCF keeps and changes: the correlation identifier of the current subscription, the fetch instruction issued for it, and the stored analytics; whether an old instruction is still served after the subscription is updated or deleted depends on that state. The model of any number of sessions cannot express the replacement of that state, so these answers are finished in the unrolled model and by the state review.

The analyst's closing note

The flow of TS 29.520 clause 4.2.2.2.2, 4.2.2.4.2 and 4.2.2.5.2 was modelled (consumer, DCCF, NWDAF) with TLS as authenticated encryption between endpoints, OAuth2 tokens bound to the requesting NF, and the notification/fetch correlation identifiers as fresh values. All eleven targets have an answer: ten proved on the model of any number of sessions with a control for each proof, and U03 answered by a proof for the notification part and by an attack on the unrolled model for the currency part. Established: the DCCF acts only for authenticated NF instances, notifications and Fetches agree with the DCCF's own identifiers, one consumer cannot obtain another's analytics, and the analytics comes from the NWDAF on the consumer's own notification path. One finding: an instruction issued before a subscription update is still answered afterwards, since nothing in the text withdraws it; a change withdrawing the identifiers on PUT was checked and removes the trace. Blocking a stronger answer: the isolation of consumers and the resource binding of a fetch correlation identifier rest on the identifiers being unguessable, while the text types them as plain strings, and the single OAuth2 scope of the API covers every operation; modelling the identifiers as guessable and the scope and audience from TS 29.501 would settle that. The checks on values that change are finished for one subscription with one update only, and a deletion, repeated updates and the fetch-instruction deadline described in the stage-2 specification are left to Tamarin+ Agent or to a further unrolling.