Workgroup: Independent Stream
Internet-Draft: draft-grothoff-taler-01
Published: 16 November 2022
Intended Status: Informational
Expires: 20 May 2023
Authors: C.G. Grothoff F.D. Dold
BFH Taler Systems SA

The 'taler' URI scheme for GNU Taler Wallet interactions

Abstract

This document defines the 'taler' Uniform Resource Identifier (URI) scheme for triggering interactions with a GNU Taler wallet.

This URI scheme allows applications to trigger interactions with the GNU Taler wallet, such as withdrawing money, making payments, receiving refunds and restoring a wallet from a backup. Applications may receive such URIs in many ways (including via NFC, QR codes, Web links or messaging applications), or might generate them internally to interact with a wallet. By having a Taler wallet handle the respective URIs, applications can integrate Taler payments without having to support the Taler protocol directly. Furthermore, by passing control to a Taler wallet process, the wallet's database with its financial data might be better protected from application failures.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 20 May 2023.

Table of Contents

1. Introduction
  1.1. Objective
  1.2. Requirements Language
2. Syntax of a 'taler' URI
3. Semantics
4. Examples
5. Tracking Taler URI Actions
  5.1. Action: withdraw
  5.2. Action: pay
  5.3. Action: refund
  5.4. Action: tip
  5.5. Action: pay-push
  5.6. Action: pay-pull
  5.7. Action: pay-template
  5.8. Action: exchange
  5.9. Action: auditor
  5.10. Action: restore
  5.11. Action: dev-experiment
6. Security Considerations
7. IANA Considerations
  7.1. URI Scheme Registration
8. Taler URI Actions
9. References
  9.1. Normative References
  9.2. Informational References
Authors' Addresses

1. Introduction

This document defines the 'taler' Uniform Resource Identifier (URI) [RFC3986] scheme for triggering interactions with GNU Taler wallets.

1.1. Objective

A 'taler' URI always instructs a GNU Taler wallet to perform a particular operation. A 'taler' URI consists of an action and optional parameters.

The interpretation of the optional parameters depends on the action.

1.2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Syntax of a 'taler' URI

This document uses the Augmented Backus-Naur Form (ABNF) of [RFC5234].

```
taler-URI = ("taler://" / "TALER://" / "taler+http://" / "TALER+HTTP:/
            action path-abempty [ "?" opts ]
action = ALPHA *( ALPHA / DIGIT / "-" / "." )
opts = opt *( "&" opt )
opt = opt-name "=" opt-value
opt-name = ALPHA *( ALPHA / DIGIT / "-" / "." / ":" )
opt-value = *pchar
```

'path-abempty' is defined in [RFC3986] in Section 3.3. 'pchar' is defined in [RFC3986], Appendix A.

3. Semantics

The action of a Taler URI identifies the operation requested from the Taler wallet. Actions are not case-sensitive. The actions are defined in the "Taler URI Actions" sub-registry, see Section 8. The path component of the URI typically provides a network address needed to locate additional information or services relevant to the requested operation. Paths are case-sensitive, but may contain case-insensitive portions, such as domain names. The query component of the URI provides immediate additional parameters or options for the operation. The query is case-sensitive. The default operation of applications that invoke a URI with the taler scheme MUST be to launch a Taler wallet (if available). If no taler URI handler is available, an application SHOULD show a QR code with the contents of the URI. If multiple Taler wallets are registered, the user SHOULD be able to choose which application to launch. This allows users with multiple wallets (each possibly with its own money) to choose which wallet to perform the operation with. An application SHOULD allow dereferencing a "taler://" URI even if the action of that URI is not registered in the "Taler URI Actions" sub-registry. Wallets seeing a "taler://" URI MUST use HTTP over TLS when talking to the respective network service. Wallets seeing a "taler+http://" URI MUST use HTTP without TLS when talking to the respective network

service. Wallets SHOULD support "taler+http://"-URIs only when run in developer or debug mode.

4. Examples

```
taler://pay/example.com/2022.268-03G33PTAY2C6T/\
  00f68430-363a-46b7-8e33-241a0e49c430?c=KKBWMSTF4AZSP8XS0FFNE9KM5M
taler://pay-push/bank.example.com/\
  3BBW6N8PVDYBRT0DERT8YYARQGFYHVQFG3WVAN1D58FRP5JG3M4G
TALER://PAY-PULL/BANK.EXAMPLE.COM/\
  WB361HXN7BZ9ND1B9YP1Y20NB4H5WS0RNM4K8AFZ5Q2VRW577BPG
```

5. Tracking Taler URI Actions

A registry of Taler URI Actions is described in Section 8. The registration policy for this registry is "Expert Review", as described in [RFC8126]. When requesting new entries, careful consideration of the following criteria is strongly advised:

1. The description clearly defines the semantics of the action and optional parameters if applicable.

2. The name states the unique name for the action that must be part of the URI.

3. The syntax defines the format of the action-specific part of the URI.

4. Relevant references are provided if they are available.

5. The chosen name is appropriate for the operation, and avoids potential to confuse users.

6. A libre software reference implementation is available.

Documents that support requests for new registry entries should provide the following information for each entry:

*Description: A description of the action, including the semantics of the path in the URI if applicable.

*Name: The name of the Taler URI action (case insensitive ASCII string, restricted to alphanumeric characters, dots and dashes)

*Syntax: summary of the syntax of the URI as a one-liner.

*Example: At least one example URI to illustrate the action.

*Contact: The contact information of a person to contact for further information

*References: Optionally, references describing the action (such as an RFC) and target-specific options.

This document populates the registry with $COUNT entries as follows (see also Section 8).

5.1. Action: withdraw

The action "withdraw" is used to trigger a bank-integrated withdrawal operation. This means that the user has been interacting with some online banking App and wants to instruct the bank to transfer money from the user's bank account into a the GNU Taler wallet. The wallet now needs to allow the user to select the GNU Taler exchange, and then ultimately provide the bank with its reserve public key and await the completion of the wire transfer.

The specific arguments of a "withdraw" action are:

*bank_host: hostname of the bank (optionally including a port number)

*bank_prefix_path: list of path components that identifies the path prefix of the bank integration API base URL

*withdrawal_uid: the unique ID of the withdrawal operation

*Name: withdraw

Syntax: taler://withdraw/{bank_host}{/bank_prefix_path}/{withdrawal_uid}

*Example: taler://withdraw/bank.example.com/wid

*Contact: N/A

*References: [this.I-D]

5.2. Action: pay

Payments are requested with the "pay" action. The parameters are a hierarchical identifier for the requested payment, and must include a hostname, order ID and session ID (which may be an empty string). Additionally, a claim token and a prefix path to be used as part of the HTTP REST API request to the hostname may be specified.

The specific arguments of a "pay" action are:

*merchant_host: hostname of the GNU Taler REST service of merchant (may optionally include a port number)

*merchant_prefix_path: list of path components that identifies the path prefix of the merchant base URL

*order_id: the order ID that the customer is asked to pay for

*session_id: the session ID under which the payment takes place

*ct: a high-entropy order "ClaimToken"

*Name: pay

Syntax: taler://pay/{merchant_host}{/merchant_prefix_path}/{order_id}/{session_id}{?c=ct}

*Example: taler://pay/merchant.example.com/42/

*Contact: N/A

*References: [this.I-D]

5.3. Action: refund

A "refund" action instructs the wallet to download information about an available refund. Wallet SHOULD consult the user about the refund and then obtain the refund for an already paid order.

The specific arguments of a "refund" action are:

*merchant_host: hostname of the merchant (possibly including a port number)

*merchant_prefix_path: list of path components that identifies the path prefix of the merchant base URL

*order_id: the order ID to check for refunds

*Name: refund

Syntax: taler://refund/{merchant_host}{/merchant_prefix_path}/{order_id}/

*Example: taler://refund/shop.example.com/42/

*Contact: N/A

*References: [this.I-D]

5.4. Action: tip

A tipping URI instructs the wallet to download information about a tip from a merchant and to ask the user to accept/decline the tip.

The specific arguments of a "tip" action are:

*merchant_host: hostname of the merchant (possibly including a port number)

*merchant_prefix_path: list of path components that identifies the path prefix of the merchant base URL

*tip_id: identifier that uniquely identifies the tip

*Name: tip

Syntax: taler://tip/{merchant_host}{/merchant_prefix_path}/{tip_id}/

*Example: taler://tip/merchant.com/FIXME

*Contact: N/A

*References: [this.I-D]

5.5. Action: pay-push

A pay-push URI instructs the wallet to ask the user about accepting a P2P payment. The wallet should download, decrypt and display the underlying contract and accept the offered money if the user agrees to the contract.

The specific arguments of a "pay-push" action are:

*exchange_host: the hostname of the exchange (possibly including a port number)

*exchange_prefix_path: list of path components that identifies the path prefix of the exchange base URL

*merge_priv: private key that grants the capability to take the money in the purse

*Name: pay-push

Syntax: taler://pay-push/{exchange_host}{/exchange_prefix_path}/{merge_priv}

*Example: taler://pay-push/exchange.example.com/
KAMRGDM8FNQ82HSBVDEH8MCAF13Q0B51P4R35RFG2CBVHKGT321G

*Contact: N/A

*References: [this.I-D]

5.6. Action: pay-pull

A pay-pull URI instructs the wallet about a request made to the user to pay an invoice (or to simply send money to another wallet). The wallet should download, decrypt and display the underlying contract and ask the user if they agree to pay the invoice.

The specific arguments of a "pay-pull" action are:

*exchange_host: the hostname of the exchange (possibly including a port number)

*exchange_prefix_path: list of path components that identifies the path prefix of the exchange base URL

*contract_priv: the private key of the peer push payment contract stored at the exchange

*Name: pay-pull

Syntax: taler://pay-pull/{exchange_host}{/exchange_prefix_path}/{contract_priv}

*Example: taler://pay-pull/exchange.example.com/
PN3W8SN6N8V1V5MTEZRPJJ2ANY8GGZMB1MBXC7NMSRXJN6MZ5SWG

*Contact: N/A

*References: [this.I-D]

5.7. Action: pay-template

A "pay-template" action instructs the wallet to ask its user to manually complete an order template and submit the information to the merchant to obtain a "pay" request. Contract fields that are not specified in the argument list must not be submitted.

Wallets MAY to support users entering all possible fields of a contract. Keys that MUST be supported at this time are the "amount" and the "summary" fields. The wallet MUST validate that the amount entered by the user is well-formed. For the amount, it is possible that the QR code already specifies the currency (e.g. "amount=CHF" or "amount=CHF:5") in which case the wallet MUST only allow the user

to enter an amount in that currency. If the amount entered by the user exceeds the wallet balance, the wallet SHOULD NOT allow the user to submit the action.

A QR code may not specify any keys for manual entry. In this case, the wallet MUST immediately submit the request (with an empty body) to the merchant to obtain a dynamically generated "taler://pay/" URI based on the template.

The specific arguments of a "pay-template" action is:

- *merchant_host: hostname of the merchant
- *merchant_prefix_path: list of path components that identifies the path prefix of the merchant base URL
- *template_id: identifier that uniquely identifies the template
- *key: possible contract detail to prompt the user for
- *value: default value to use for the respective key
- *Name: pay-template
- *Syntax: taler://pay-template/{merchant_host}{/merchant_prefix_path*}/{template_id}[?key[=value]}{&key[=value]}*
- *Example: taler://pay-template/merchant.example.com/FEGHYJY48FEGU6WETYIOIDEDE2QW3OCZVY?amount=KUDOS:5
- *Contact: N/A
- *References: [this.I-D]

5.8. Action: exchange

An "exchange" action instructs the wallet to display a prompt to the user, asking the user to confirm/decline adding the exchange to the list of trusted exchanges.

The specific arguments of an "exchange" action are:

- *exchange_host: hostname of the exchange (possibly including a port number)
- *exchange_prefix_path: list of path components that identifies the path prefix of the exchange base URL

*exchange_pub: the public key of the exchange

*Name: exchange

Syntax: taler://exchange/{exchange_host}{/exchange_prefix_path}/
{exchange_pub}

*Example: taler://exchange/exchange.example.com/
ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789ABCDEFGHIJKLMNOP

*Contact: N/A

*References: [this.I-D]

5.9. Action: auditor

An "auditor" action instructs the wallet to display a prompt to the user, asking the user to confirm/decline adding the auditor to the list of trusted auditors.

The specific arguments of an "auditor" action are:

*auditor_host: the hostname of the auditor (possibly including a port number)

*auditor_prefix_path: list of path components that identifies the path prefix of the auditor base URL

*auditor_pub: the public key of the auditor

*Name: auditor

Syntax: taler://auditor/{auditor_host}{/auditor_prefix_path}/
{auditor_pub}

*Example: taler://auditor/auditor.example.com/
ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789ABCDEFGHIJKLMNOP

*Contact: N/A

*References: [this.I-D]

5.10. Action: restore

A "restore" action instructs the wallet to restore a wallet backup and merge it into its current state.

The specific arguments of a "restore" action are:

*sync_rootkey: Root sync key of the wallet, used to derive the symmetric key to encrypt the backup with individual providers.

*sync_provider_list: Comma-separated list of provider http or https URLs. If no scheme part is specified, https is assumed. Each URL is URI-encoded for all characters except "A-Z a-z 0-9 - _ . ! ~ * ' ( )" (matching the HTML5 encodeURIComponent).

*Name: restore

*Syntax: taler://auditor/{sync_rootkey}/{sync_provider_list}

*Example: taler://restore/backup.example.com/GJKG23V4ZBHEH45YRK7TWQE8ZTY7JWTY5094TQJSRZN5DSDBX8E0/prov1.example.com,prov2.example.com

*Contact: N/A

*References: [this.I-D]

5.11. Action: dev-experiment

An "dev-experiment" action instructs the wallet to simulate a particular error scenario. This action can be used to test the user interface. Wallets that are not in developer mode should not run the specified action and instead inform the user that "dev-experiment" actions are only supported in developer mode.

The specific arguments of a "dev-experiment" action are:

*name: specifies the specific type of dev experiment

*Name: dev-experiment

*Syntax: payto://dev-experiment/{name}

*Example: payto://dev-experiment/xxx

*Contact: N/A

*References: [this.I-D]

6. Security Considerations

Interactive applications handling the taler URI scheme MUST NOT initiate any unsafe payment operations prior review and confirmation from the user, and MUST take measures to prevent clickjacking [HMW12].

The authentication/authorization mechanisms and transport security services used to process a payment encoded in a taler URI are handled by the application and are not in scope of this document.

7. IANA Considerations

IANA maintains a registry called the "Uniform Resource Identifier (URI) Schemes" registry.

7.1. URI Scheme Registration

IANA maintains the "Uniform Resource Identifier (URI) Schemes" registry that contains an entry for the 'taler' URI scheme. IANA is requested to update that entry to reference this document when published as an RFC.

8. Taler URI Actions

This document specifies a list of Taler URI Actions. It is possible that future work will need to specify additional actions. The GNUnet Assigned Numbers Authority (GANA) [GANA] operates the "taler-uri-actions" registry to track the following information for each payment target type:

- Name: The name of the action (case insensitive ASCII string, restricted to alphanumeric characters, dots and dashes)

- Contact: The contact information of a person to contact for further information

- References: Optionally, references describing the payment target type (such as an RFC) and target-specific options, or references describing the payment system underlying the payment target type.

The entries that have been made for the "taler-uri-actions" defined in this document are as follows:

Name	Contact	Reference
pay	N/A	[This.I-D]
withdraw	N/A	[This.I-D]
refund	N/A	[This.I-D]
tip	N/A	[This.I-D]
pay-pull	N/A	[This.I-D]
pay-push	N/A	[This.I-D]
pay-template	N/A	[This.I-D]
exchange	N/A	[This.I-D]
auditor	N/A	[This.I-D]
restore	N/A	[This.I-D]
dev-experiment	N/A	[This.I-D]

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[RFC5234] Crocker, D., Ed., Overell, P., and RFC Publisher, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/info/rfc5234>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

9.2. Informational References

[GANA] GNUnet e.V., "GNUnet Assigned Numbers Authority (GANA)", April 2020, <https://gana.gnunet.org/>.

[HMW12] Huang, L.S., Moshchuk, A., Wang, H.J., Schecter, S., and C. Jackson, "Clickjacking: Attacks and Defenses", January 2012, <https://www.usenix.org/system/files/conference/usenixsecurity12/sec12-final39.pdf>.

Authors' Addresses

Christian Grothoff
BFH
Höheweg 80
CH-2501 Biel/Bienne
Switzerland

Email: christian.grothoff@bfh.ch

Florian Dold
Taler Systems SA
7, rue de Mondorf
L-5421 Erpeldange
Luxembourg

Email: dold@taler.net