Applications Area Working Group
Internet Draft
Intended status: Informational
Expires: July 2016

D. Warden
Dell Products LP
I. Iordanov
Undatech

January 10, 2016

The "vnc" URI Scheme
draft-warden-appsawg-vnc-scheme-08.txt

Abstract

Virtual Network Computing (VNC) software provides remote desktop functionality. This document describes a Uniform Resource Identifier (URI) scheme enabling the launch of VNC clients from other applications. The scheme specifies parameters useful in securely connecting clients with remote hosts.

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), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

This Internet-Draft will expire on May 18, 2016.

Table of Contents

1. Introduction

Virtual Network Computing (VNC) clients are used to support remote desktop connectivity based on the Remote Framebuffer (RFB) Protocol [RFC6143]. It is often desirable to integrate such functionality

with other software. However, the lack of a standard method for specifying VNC client parameters has limited such integration.

The "vnc" Uniform Resource Identifier (URI) scheme specified in this document facilitates the launch of VNC clients from applications in browser-based, desktop, and mobile environments. Using this scheme, users and application vendors will be able to integrate remote desktop capabilities without being tied to a particular client.

Remote desktop clients often store connection profiles in a local connection database. By associating connections specified in a URI with those stored in a database, client-specific options can be automatically applied to a connection launched from another application, even when that application is unaware of those options.

Connections to VNC servers are often secured using mechanisms including Transport Layer Security/Secure Sockets Layer (TLS/SSL) tunneling [RFC5246] and Secure Shell (SSH) [RFC4251] tunneling which are outside the scope of the RFB protocol. Defining the behavior of these client-integrated security options enables their use with "vnc" URIs.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119 [RFC2119].

In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying RFC-2119 significance.

2. The "vnc" URI Scheme

2.1. URI Scheme Syntax

The normative syntax of the "vnc" URI is defined in the <vnc-uri> rule in the following syntax specification. This specification uses the Augmented Backus-Naur Form (BNF) as described in [RFC5234]. The "vnc" URI conforms to the generic URI syntax specified in [RFC3986]. The <userinfo>, <host>, <port>, <unreserved>, and <pct-encoded> rules are defined in [RFC3986].

```
vnc-uri = "vnc://" [ userinfo "@" ] [ host [ ":" port ] ]

        [ "?" [ vnc-params ] ]
```

```
vnc-params = param "=" value *("&" param "=" value) ["&"]

param = 1*( param-char )

value = *( param-char )

param-char = unreserved / pct-encoded / unreserved-symbols

unreserved-symbols = ":" / "/" / "@" / "!" / "$" / "'"

                   / "(" / ")" / "*" / "," / ";"
```

The "?", "=", and "&" characters are used to delimit VNC parameters and must be percent-encoded when representing a data octet as specified in [RFC3986]. Within the <vnc-params> portion of a "vnc" URI the <unreserved-symbols> do not have special meaning and need not be percent-encoded when representing a data octet.

A "vnc" URI has the general form:

```
vnc://host:port?param1=value1&param2=value2...
```

The host information and each parameter value specify information used in establishing or operating the remote desktop session as specified in Section 2.1.1.

For example:

```
vnc://10.0.0.1:5901?VncPassword=secret&SecurityType=2
```

Indicates a vnc connection to the host at IP "10.0.0.1" on port "5901" with VNC password "secret" using the VNC Authentication security type.

2.1.1. URI Parameters

A description of host information and URI parameters is provided in this section. Information on the constraints of various data types is provided in Section 2.1.2. All parameters are considered optional, however a client will not be able to connect without sufficient information.

A parameter without a specified default value indicates that no default value is implied by this URI scheme, however VNC clients can apply implementation dependent default behaviors otherwise consistent with this document.

The <userinfo> value is deprecated and processed only in an implementation-specific manner. The <userinfo> component MUST NOT be generated in an environment where a client supporting an updated URI format is expected to be available. When processing a URI value from an untrusted source, VNC clients SHOULD alert the user in order to mitigate the risk that the URI is constructed to obscure the identity of the remote host unless the URI can be validated or backwards compatibility considerations make an alert impractical.

The <host> and <port> values in the "vnc" URI specify the address of the VNC server on the remote host:

Name	Type	Description	Default
host	string	VNC server hostname or IP	none
port	ushort	VNC server port	5900

The "vnc" URI parameter values specify remote desktop connection or session properties, including aspects of client operation, usability, and security as specified in the table below:

Name	Type	Description	Default
ConnectionName	string	Name of connection profile	none
VncUsername	string	VNC server username	none
VncPassword	string	VNC server password	none
SecurityType	enum <rfbsec>	RFB security type used	none
ChannelType	enum <chan>	Connection channel type	none
SshHost	string	SSH server hostname or IP	<host>
SshPort	ushort	SSH server port	22
SshUsername	string	SSH username	none
SshPassword	string	SSH password	none

IdHashAlgorithm	enum <idhash>	Hash algorithm used with "IdHash" parameter	none
IdHash	string <hex>	Expected hash of remote public key or certificate	none
ColorLevel	enum <clevel>	Client color depth/mode	none
ViewOnly	boolean	Client is view only	false
SaveConnection	boolean	Store connection info	none

o ConnectionName, SaveConnection

The "ConnectionName" is used to identify a connection profile in both the launching application and VNC client. Profiles are applied as described in Section 2.2.2. If omitted, the client MAY generate a name based on the host, port, and/or other parameters. The VNC client MAY normalize the name as required.

If true, "SaveConnection" indicates a connection profile should be created or updated and stored in the client connection database. If false, no profile should be updated or persisted.

o VncUsername, VncPassword, SecurityType

The SecurityType parameter indicates which RFB security type applies to the connection. RFB security types are recorded in the IANA "Remote Framebuffer Security Types" registry created by [RFC6143]. The VNC client will use this information to determine which parameters are required and establish the connection.

VNC clients can sometimes automatically negotiate a security type with a server. However, in addition to controlling the security negotiation, specifying the security type also allows for a client to prompt in advance for necessary security parameters. Parameters may take time to enter on mobile clients, and could otherwise result in timeouts and/or security lockouts. If the specified type is not supported by the server, an error SHOULD be indicated as described in Section 2.2.1.

The "VncUsername" and "VncPassword" are used when applicable to authenticate to the VNC server using the specified "SecurityType". Since passwords often contain arbitrary characters, they will often require percent encoding.

- ChannelType

  The channel type specifies the transport stream used to carry connection data. This allows a client to initiate a connection using a secure transport protocol such as SSH prior to connecting to the VNC server socket. Use of this value in the context of the "Integrated SSH" and "Secure Tunnel" channel types is provided in Section 2.3.

- SshHost, SshPort, SshUsername, SshPassword

  The SSH parameters are intended for use with the "Integrated SSH" channel type described in Section 2.3.1. These parameters can also be used with any future SSH-based channel types. Since passwords often contain arbitrary characters, they will often require percent encoding.

- IdHashAlgorithm, IdHash

  The "IdHashAlgorithm" and "IdHash" values are used to verify the expected identity of the remote system based on its public key or certificate. Use of these values in the context of the "Integrated SSH" and "Secure Tunnel" channel types is provided in Section 2.3.

- ColorLevel

  The "ColorLevel" parameter specifies the color model to use for data transfer and display as specified in Section 2.1.2. If the requested color model is unsupported, the behavior is implementation dependent.

- ViewOnly

  If true, the VNC client SHOULD operate in a display-only mode and refrain from sending input data including KeyEvent, PointerEvent, and ClientCutText messages specified in Section 7.5 of [RFC6143] unless this mode is unsupported by the client.

Parameter names SHOULD be provided in the case specified in this document, however for compatibility clients SHOULD accept parameters

in a case-insensitive manner. Values SHALL be interpreted in a case-sensitive manner, unless otherwise noted.

Additional parameters likely to be useful with multiple VNC clients can be added to the "VNC URI Parameters" registry as specified in Section 4.6 of this document. Individual clients MAY support parameters specific to that client. VNC Clients supporting application-specific parameters SHOULD include a distinguishing prefix within the parameter name, such as the name of the application package specified in source code except when precluded by compatibility constraints. For example:

```
vnc://?com.dell.vncclient.ScreenMode=2&
```

It can also be expected that clients will maintain backward compatibility with legacy URI formats and parameters.

2.1.2. Data Types

"vnc" URIs can be percent-encoded as specified in [RFC3986] and MUST be decoded. After decoding, the following type constraints and semantics apply:

- string

  Values of "string" type are UTF-encoded strings as specified in [RFC3629].

  The "string<hex>" subtype used in the "IdHash" consists of colon-delimited ":" octets displayed in hexadecimal. For example:

  ```
  5D:D2:39:57
  ```

  Comparison of "string<hex>" values SHALL be case-insensitive, however the uppercase notation is preferred for readability.

- enum

  The "enum" types consist of specific enumerated subtypes and are represented by their decimal value.

  The "enum<rfbsec>" values represent an RFB security type included in the IANA "Remote Framebuffer Security Types" registry created by [RFC6143].

"enum<chan>" values represent connection channel types listed in the "VNC URI Connection Channel Types" registry created by Section 4.4 of this document. Initial values are:

Value	Description
1	Standard TCP
23	Secure Tunnel
24	Integrated SSH

The Standard TCP channel type represents a generic TCP connection. The Secure Tunnel and Integrated Secure Shell (SSH) [RFC4252] channel types are described in Section 2.3.

Values of the "enum<idhash>" parameter represent secure hash algorithms in the "VNC URI Hash Algorithms" registry created by Section 4.5 of this document. The initial values include:

Value	Description
1	MD5
2	SHA1
4	SHA256

The MD5 algorithm is described in [RFC1321]. The SHA1 and SHA256 algorithms are described in [SHS].

Values of the "enum<clevel>" subtype represent a color level. In the table below, the columns have the meaning specified in Section 7.4 of [RFC6143]:

BPP = bits-per-pixel
TC = true-color-flag
RM = red-max
GM = green-max
BM = blue-max
RS = red-shift
GS = green-shift
BS = blue-shift

The values are:

Value	Description	BPP	Depth	TC	RM	GM	BM	RS	GS	BS
1	Black and White	8	3	t	1	1	1	2	1	0
2	Greyscale	8	6	t	3	3	3	4	2	0
3	8 Colors	8	3	t	1	1	1	2	1	0

Value	Description	BPP	Depth	TC	RM	GM	BM	RS	GS	BS
4	64 Colors	8	6	t	3	3	3	4	2	0
5	256 Colors	8	8	t	7	7	3	0	3	6
6	16-bit Color	16	16	t	31	63	31	11	5	0
7	24-bit Color	32	24	t	255	255	255	16	8	0
8	30-bit Color	32	30	t	1023	1023	1023	0	10	20

A value of "t" indicates the true-color-flag should be set. The big-endian-flag should be set as required for the system.

- ushort

  The "ushort" values represent unsigned 16-bit integers expressed in decimal digits with value between 0-65535 inclusive.

- boolean

  "boolean" values represent conditions that are true or false and are represented as either "true" or "false" respectively. For maximum compatibility, clients SHOULD accept the value 1 as representing true values and 0 as representing false values. Clients SHOULD perform parsing of "boolean" values in a case-insensitive manner.

An example "vnc" URI including several of these data types is:

```
vnc://localhost:5900?ConnectionName=Server&SecurityType=2&
   IdHash=0D:3A:72:08:57:EA:4D:30&SaveConnection=false&
```

Note the above example should be considered to be a contiguous string without line breaks or whitespace and is broken into multiple lines in this document for readability.

2.2. Processing URIs

Conceptually, a VNC URI supports only a "VIEW" operation, indicating the user wishes to view the remote desktop accessible via the URI reference.

In general, when a VNC client receives a "vnc" URI it will initiate an RFB protocol remote desktop connection using the specified host information and parameter values. Initiating the connection using a connection channel mechanism such as those specified in Section 2.3 might require processing prior to establishing the RFB connection. A client MAY attempt to automatically discover or negotiate appropriate connection channel, security, or other parameter values.

The process for negotiating security types is specified in [RFC6143]. Supported connection channels could be discovered by testing channel types to detect when a channel is successfully established. To best integrate with other applications the VNC client SHOULD initiate the connection with minimal or no user intervention, whenever sufficient information is available and adequate security is preserved.

Host information and parameter values may be provided through connection profiles. When a parameter value is not available from either a URI or a connection profile described in Section 2.2.2, the default value specified in Section 2.1.1 SHOULD be applied. If available parameters are not sufficient to establish a connection, the VNC client SHOULD present a session initiation data-entry screen.

2.2.1. Error Handling

In a typical interactive environment, if an error prevents a session from being established, the VNC client presents an error message to the user. When the message is acknowledged, the console application can show a session initiation data-entry screen populated with available session parameters or it can terminate. If an error occurs after a session is successfully established that terminates the connection, the VNC client presents a termination notification to the user. When the termination notification is acknowledged, the client can present a reconnection prompt or terminate.

When an error occurs in a dedicated environment (such as a kiosk system), the system can transmit an alert to the remote operator, record a log entry, and execute appropriate fallback behavior such as automatically attempting to reestablish a session or displaying a generic message requesting servicing.

2.2.2. Connection Profile Matching

VNC clients MAY store remote desktop session settings in connection profiles. If the client is able to uniquely identify and associate a connection request with a connection profile based on the "ConnectionName" parameter value, remote host IP address, or hostname/fully-qualified domain name, the VNC client SHOULD apply profile values for those settings which do not have values supplied in the "vnc" URI. When profile data is unavailable, the VNC client MAY apply global application defaults for settings not supplied in the URI and for which the scheme does not specify a default value. The VNC client MUST NOT override supplied parameters with profile values or global defaults.

When the "SaveConnection" parameter value is true, within the VNC client a connection profile SHOULD be created or updated with the values supplied in the "vnc" URI. Profile updates and storage should be consistent with the recommendations in Section 3.4.

2.3. Connection Channel Types

2.3.1. The "Integrated SSH" Channel Type

The "Integrated SSH" channel type establishes an SSH connection to a host, authenticates with SSH password authentication, establishes a secure tunnel to the VNC host/port, and then connects to the VNC server using a supported "SecurityType". The secure tunnel will provide encryption and data integrity, while verifying the public key authenticates the server. The SSH architecture is specified in [RFC4251]. The steps are detailed below:

1. The VNC client initiates a transport-level connection to the "SshHost" on the "SshPort" specified in the parameter values with a key exchange as described in [RFC4253].

2. When the VNC client receives the server key (or certificate), the hash of the key (or certificate) is computed using the algorithm corresponding to the "IdHashAlgorithm" parameter value and compared with the expected "IdHash" value (if available). If the certificate hash cannot be verified, the client alerts the user or operator. In a typical interactive environment, the alert provides the remote system's identifying information including the hash value and allows the user to terminate the connection. The alert could allow the user to accept the key and continue establishing the connection. In a dedicated environment (such as a kiosk system), the system can transmit an alert to the remote operator, record a log entry, and execute appropriate fallback behavior such as displaying a generic message requesting servicing.

3. The SSH client authenticates the user using the "SshUsername" and "SshPassword" parameter values according to the "password" authentication mechanism described in [RFC4252].

4. The SSH client opens a TCP/IP channel as specified in [RFC4254] from the local system to the system indicated by the <host> and <port> information values.

5. The VNC client establishes a RFB connection to the VNC server over the channel and authenticates using the "SecurityType" as described in [RFC6143] or other reference.

The VNC client MAY establish the connection described in this section using an external SSH client, by launching the client and then connecting to a secure tunnel created between a local port and the VNC server.

If the VNC client is supplied with additional parameters outside the scope of this document, it MAY perform a variation of these steps consistent with the underlying protocols, for example by using "publickey" SSH client authentication [RFC4252] or providing another form of authentication to the VNC server. The specific negotiation of SSH parameters such as cipher suite configuration is outside the scope of this document.

Many SSH clients present key hashes using MD5 and it can be expected that launching applications will specify the hash be displayed in the manner its users are familiar with.

For compatibility, when the "SecurityType" parameter value is "Integrated SSH" (24), a VNC client MUST treat the value as a request to use "Integrated SSH" as the "ChannelType". However, this value SHOULD NOT be supplied for the "SecurityType" parameter unless required for backward compatibility as the channel is established prior to connecting to the server and is not consistent with the negotiation of other security types.

2.3.2. The "Secure Tunnel" Channel Type

The "Secure Tunnel" channel type establishes a TLS connection with a remote server using certificate authentication, over which a connection to the VNC server is established using a supported "SecurityType". The secure tunnel will provide encryption and data integrity, while verifying the certificate authenticates the server. The TLS protocol is specified in [RFC5246]. The steps are detailed below:

1. The VNC client initiates the TLS Handshake Protocol with a system indicated by the <host> and <port> information values.

2. When the server certificate is received, the hash of the key certificate is computed using the algorithm corresponding to the "IdHashAlgorithm" parameter value and compared with the expected "IdHash" value (if available). If the certificate hash cannot be verified, the client alerts the user or operator. In a typical interactive environment, the alert provides the remote system's identifying information and allows the user to terminate the connection. The alert could allow the user to accept the key and continue establishing the connection. In a dedicated environment (such as a kiosk system), the system can transmit an alert to the remote operator, record a log entry, and execute appropriate fallback behavior such as displaying a generic message requesting servicing.

   When providing identifying information of a host identified by an X509 certificate [RFC5280], the certificate subject, issuer, validity period, and certificate hash is typically included. The VNC client MAY verify the validity of the certificate. If the validity of a certificate is not confirmed, the alert includes a statement indicating such information has not been verified.

3. The client finishes establishing the TLS tunnel.

4. The VNC client establishes a RFB connection to the VNC server over the channel and authenticates using the "SecurityType" as described in [RFC6143] or other reference.

If the VNC client is supplied with additional parameters, it MAY perform a variation of these steps consistent with the underlying protocols, for example by providing another form of authentication to the VNC server. The negotiation of specific TLS parameters such as cipher suite configuration is outside the scope of this document.

The TLS protocol provides backwards compatibility with SSLv3, however due to known security flaws it SHOULD NOT be used.

For compatibility, when the "SecurityType" parameter value is "Secure Tunnel" (23) a VNC client MUST treat the value as a request to use "Secure Tunnel" as the "ChannelType". However, this value SHOULD NOT be supplied for the "SecurityType" parameter unless required for backward compatibility as the channel must be established prior to connecting to the server and is not consistent with the negotiation of other security types.

3. Security Considerations

General security concerns involving URI schemes are discussed in [RFC3986]. In implementing support for the "vnc" URI scheme, areas for particular consideration include application trust, URI transmission, host identification, and connection database security.

Remote desktop connectivity requires the transmission of security credentials, which could be included in a URI. If those credentials are not kept secure, an attacker can gain access to any systems using those credentials. Host addresses and connection parameters might also be considered sensitive, as such information can be used in planning an attack.

URIs can also contain host identification information. It is important to securely identify the remote host system to which a connection is established. If a user connects to an attacker's system, user data, including credentials, can be exposed.

Note that the RFB protocol itself may not encrypt data. To protect data in transit, RFB should be tunneled over TLS [RFC5246], SSH [RFC4251], or another secure protocol.

Some VNC systems can be used without authentication. To protect the remote host, strong passwords or other authentication mechanisms need to be used.

3.1. Application Trust

A malicious application receiving VNC credentials via URI or other means can obviously misuse those credentials. To protect against this, users should only install applications from trusted sources. The integrity of application packages can be verified through digital signatures.

Applications launching VNC clients can elect to launch only particular trusted clients, and can specify those clients through platform-specific mechanisms. Package integrity can be verified programmatically by querying the package manager for digital signatures or other platform-specific means.

The risk to a VNC client from a launching application is generally much lower, since the launching application will not receive credentials or data from the client. A VNC client can verify its caller thorough platform-specific means.

VNC clients ought not to accept potentially destructive parameters from untrusted launching applications without explicit user confirmation. For example, a client-specific parameter that runs an arbitrary command upon establishing a SSH connection used for VNC tunneling is potentially destructive and high risk.

3.2. URI Transmission

Within a mobile or desktop environment, application launch will typically involve in-memory URI data transmission facilitated and secured by the operating system.

If sensitive URI information is exchanged across a network, for example by providing a list of connection URIs in a web page, the data needs to be encrypted in transit and only be accessible to authorized users.

When an application detects potentially sensitive information in a VNC URI, it needs to be handled securely or discarded. In particular, the URI data that is persisted needs to be encrypted as described in Section 3.4.

Applications that process URIs in a generic way, such as web browsers, might not detect that sensitive information is contained in a URI and could cache or store that information insecurely. It is advisable to avoid including credentials and other sensitive information in URIs that are likely to be processed in a generic way unless such caching and storage is disabled or otherwise secured.

3.3. Host Identification

In the absence of verifiable host identification, a VNC client application is vulnerable to spoofing and man-in-the-middle attacks that capture VNC or host OS credentials and user data. To prevent such attacks, administrators SHOULD secure their VNC communications with TLS [RFC5246] or SSH [RFC4251] tunnels or other connection mechanisms identifying remote hosts via certificate or public key. VNC clients MUST verify the respective certificates or public keys to confirm the remote host's identity.

An application launching a VNC client via URI MAY provide a certificate hash or public key hash identifying the remote host. VNC clients maintaining a connection database can also store certificate or public key data suitable for validating a host's identity.

If connecting to a system identified by certificate or public key and a remote system ID hash cannot be matched to available identifying data, the VNC client needs to alert the user or operator. In a typical interactive environment, the alert will provide the remote system's identifying information and allow the user to terminate the connection. The alert can allow the user to accept the information and continue establishing the connection. In a dedicated environment (such as a kiosk system), the system can transmit an alert to the remote operator, record a log entry, and execute appropriate fallback behavior such as displaying a generic message requesting servicing.

When providing identifying information of a host identified by an X509 certificate [RFC5280], the certificate subject, issuer, validity period, and certificate hash needs to be included. The VNC client can verify the certificate validity. If the validity of a certificate is not determined, the alert needs to include a statement indicating such information has not been verified.

Identifying information of a host identified by public key, such as the endpoint of an SSH connection using a raw key, needs to include a hash of the key.

3.4. Connection Database Integrity

A VNC client application and/or launching application can maintain a connection database containing remote host information, credentials, and/or connection parameters. Applications storing credentials need to ensure they are stored in an encrypted format with a decryption process requiring user-supplied or device-specific data. If supported, it is advisable for applications to have a setting disabling storage of credentials.

If available, the VNC client connection database can store certificate or public key data used to verify host identification. To prevent a malicious URI from overriding the database, if identification information in the URI conflicts with information in the database, the user or operator needs to be alerted. In a typical interactive environment, the user can be prompted to accept the new information prior to updating the database.

4. IANA Considerations

The "vnc" scheme should be registered in the URI schemes registry.

IANA "Remote Framebuffer Security Types", "VNC URI Connection Channel Types", "VNC URI ID Hash Algorithms", and "VNC URI Parameters" registries will support elements of the scheme.

4.1. "vnc" Scheme

IANA is asked to add the "vnc" scheme to the "Uniform Resource Identifier (URI) Schemes" registry with description "Remote Framebuffer Protocol" and reference to this document. A registration template is provided in Appendix A.

The IANA schemes registry is currently located at:

http://www.iana.org/assignments/uri-schemes/uri-schemes.xhtml

4.2. Remote Framebuffer Security Types

This document references the existing IANA "Remote Framebuffer Security Types" registry in specifying security type options. RFB security types are supported in "vnc" URIs.

Security mechanisms integrated with VNC clients might need to alter the process by which a connection is established prior to the security handshake described in Section 7.1.2 of RFC6143. Such mechanisms should be reflected in the "VNC URI Connection Chanel Types" registry described in Section 4.4 of this document rather than the "Remote Framebuffer Security Types" registry as their use cannot be negotiated by the mechanism specified in RFC6143.

Exceptions can be made for backwards compatibility. IANA is asked to update the "Secure Tunnel" and "Integrated SSH" security types listed for compatibility to refer to this document.

4.3. VNC URI Group

IANA is asked to create a "VNC URI" Group. This group will contain application-level URI related registries distinct from those used by the RFB protocol itself.

4.4. VNC URI Connection Channel Types

IANA is asked to create a "VNC URI Connection Channel Types" registry within the "VNC URI" group. The registry should include

Value, Description, and Reference columns. The initial contents of the registry are described in this document. The values of the "Secure Tunnel" and "Integrated SSH" types are copied from the RFB Security Types registry. They should be:

Value	Description	Reference
0	Reserved	(this document)
1	Standard TCP	(this document)
23	Secure Tunnel	(this document)
24	Integrated SSH	(this document)

The maximum acceptable value is 2,147,483,647.

Future assignments to this registry should be made through the "Expert Review" process described in [RFC5226]. Experts reviewing VNC URI Connection Channel Types should verify the proposed channel type description is clear, and the channel type does not conflict with existing channel types that are registered or in widespread use.

4.5. VNC URI ID Hash Algorithms

IANA is asked to create a "VNC URI ID Hash Algorithms" registry within the "VNC URI" group. The registry should include Value, Description, and Reference columns.

The initial hash algorithms specified are a subset of the algorithms contained in the "TLS HashAlgorithm Registry". The initial contents of the registry should be:

Value	Description	Reference
0	Reserved	(this document)
1	MD5	(this document)
2	SHA1	(this document)
4	SHA256	(this document)

The maximum acceptable value is 2,147,483,647.

Future assignments to this registry should be made through the "Expert Review" process described in [RFC5226]. Experts reviewing VNC URI ID Hash Algorithms should verify the proposed parameter description is clear, and the parameter does not conflict with existing parameters that are registered or in widespread use.

4.6. VNC URI Parameters

IANA is asked to create a "VNC URI Parameters" registry within the "VNC URI" group.

The initial contents are described in this document. They should be:

Name	Description	Reference
ConnectionName	Name of connection profile	(this document)
VncUsername	VNC server username	(this document)
VncPassword	VNC server password	(this document)
SecurityType	RFB security type used	(this document)
ChannelType	Connection channel type	(this document)
SshHost	SSH server hostname or IP	(this document)
SshPort	SSH server port	(this document)
SshUsername	SSH username	(this document)
SshPassword	SSH password	(this document)
IdHashAlgorithm	Hash algorithm used with "IdHash" parameter	(this document)
IdHash	Expected hash of remote public key or certificate	(this document)
ColorLevel	Client color depth/mode	(this document)
ViewOnly	Client is view only	(this document)
SaveConnection	Store connection info	(this document)

Future assignments to these registries should be made through the "Expert Review" process described in [RFC5226]. Experts reviewing VNC URI parameters should verify the proposed parameter name and description are clear, and the parameter does not conflict with existing parameters that are registered or in widespread use.

5. References

5.1. Normative References

[RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC4251] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Protocol Architecture", RFC 4251, January 2006.

[RFC4252] Ylonen, T. and C. Lonvick, "The Secure Shell (SSH) Authentication Protocol", RFC 4252, January 2006.

[RFC4253] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Transport Layer Protocol", RFC 4253, January 2006.

[RFC4254] Ylonen, T. and C. Lonvick, "The Secure Shell (SSH) Connection Protocol", RFC 4254, January 2006.

[RFC5234] Crocker, D., Ed., and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet x.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.

[RFC6143] Richardson, T., and J. Levine, "The Remote Framebuffer Protocol", RFC 6143, March 2011.

[SHS] NIST FIPS PUB 180-2, "Secure Hash Standard", National Institute of Standards and Technology, U.S. Department of Commerce, August 2002.

5.2. Informative References

[RFC7595] Thaler, D., Ed., Hansen, T., and T. Hardie, "Guidelines and Registration Procedures for URI Schemes", BCP 35, RFC 7595, June 2015.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[X.509] ITU-T Recommendation X.509 (2005) | ISO/IEC 9594-8:2005, Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks.

6. Acknowledgments

Dominic Parkes and the staff of RealVNC Ltd. graciously reviewed this document and provided constructive comments.

This document was prepared using 2-Word-v2.0.template.dot.

Appendix A. "vnc" URI Template

This template is provided for registration of the VNC URI in the IANA URI schemes registry as specified in [RFC7595].

URI Scheme name: vnc

Status: Permanent

URI scheme syntax: See Section 2 of this document.

Scheme semantics: See Section 2 of this document.

Encoding considerations: See Section 2 of this document.

Applications/protocols that use this URI scheme name: Virtual Network Computing (VNC) remote desktop applications use vnc URIs. VNC applications use the Remote Framebuffer (RFB) protocol.

Interoperability considerations: Legacy software applications respond to vnc URIs in different ways and may fail to behave as expected. It is advisable to test vnc URIs with specific applications or consult application-specific documentation.

Security considerations: See Section 3 of this document.

Contact: IESG <iesg@ietf.org>.

Author/Change Controller: See the Authors of this document. Change control is through the IESG on behalf of the IETF <iesg@ietf.org>.

References: This document.

Authors' Addresses

David Warden
Dell Products LP
200 Dell Way
Round Rock, Texas 78682
U.S.A.

Phone: 512-728-0380
Email: David_Warden@dell.com
URI: http://www.dell.com

Iordan Iordanov
Undatech
260 Scarlet Road, Apt. 503
Toronto, ON M6N 4X6
CANADA

Email: iiordanov@gmail.com