The "vnc" URI Scheme
draft-warden-appsawg-vnc-scheme-05.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 March 30, 2016.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.
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

Table of Contents

1. Introduction .................................................................................. 2
   1.1. Requirements Language .................................................. 3
2. The "vnc" URI Scheme .................................................................. 3
   2.1. URI Scheme Syntax ...................................................... 3
      2.1.1. URI Parameters ................................................. 4
      2.1.2. Data Types ....................................................... 8
   2.2. Processing URIs ........................................................... 10
      2.2.1. Error Handling .................................................. 11
      2.2.2. Connection Profile Matching ............................... 11
   2.3. Connection Channel Types ............................................... 11
      2.3.1. The "Integrated SSH" Channel Type ...................... 11
      2.3.2. The "Secure Tunnel" Channel Type ....................... 13
3. Security Considerations ............................................................. 14
   3.1. Application Trust .......................................................... 14
   3.2. URI Transmission ......................................................... 15
   3.3. Host Identification ....................................................... 15
   3.4. Connection Database Integrity ....................................... 16
4. IANA Considerations ................................................................. 16
   4.1. "vnc" Scheme .............................................................. 17
   4.2. Remote Framebuffer Security Types ................................ 17
   4.3. VNC Connection Channel Types .................................... 17
   4.4. VNC ID Hash Algorithms .......................................... 18
   4.5. VNC URI Parameters .................................................. 18
5. References ................................................................................... 19
   5.1. Normative References .................................................. 19
   5.2. Informative References ................................................. 20
6. Acknowledgments ........................................................................ 20

Appendix A. "vnc" URI Template ...................................................... 22
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?parm1=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 VNC security.

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.

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.

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   |          |
| 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  |
+---------------+---------+-----------------------------+----------+
<p>| ConnectionName| string  | Name of connection profile  |          |
| VncUsername   | string  | VNC server username         |          |
| VncPassword   | string  | VNC server password         |          |
| SecurityType  | enum    | VNC security type used      |          |
|               | &lt;vncsec&gt;                             |          |
| ChannelType   | enum    | Connection channel type     |          |
|               | &lt;chan&gt;  |                             |          |
| SshHost       | string  | SSH server hostname or IP   | &lt;host&gt;   |
| SshPort       | ushort  | SSH server port             | 22       |
| SshUsername   | string  | SSH username                |          |
| SshPassword   | string  | SSH password                |          |
| IdHashAlgorithm| enum   | Hash algorithm used with    |          |
|               | &lt;idhash&gt; &quot;IdHash&quot; parameter         |          |
| IdHash        | string  | Expected hash of remote     |          |
|               | &lt;hex&gt;    public key or certificate |          |</p>
<table>
<thead>
<tr>
<th>ColorLevel</th>
<th>enum</th>
<th>Client color depth/mode</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&lt;clevel&gt;</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ViewOnly</th>
<th>boolean</th>
<th>Client is view only</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>true</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SaveConnection</th>
<th>boolean</th>
<th>Store connection info</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>true</td>
<td></td>
</tr>
</tbody>
</table>

- **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.

- **VncUsername**, **VncPassword**, **SecurityType**
  
  The SecurityType parameter indicates which RFB security type applies to the connection. VNC 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.
o 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.

o 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.

o 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.

o 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.

o 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].

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 4.5 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. 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

As "vnc" URIs may 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 index value.

  The "enum<vncsec>" values represent a VNC 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 Connection Channel Types" registry created by Section 4.3 of this document. Initial values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Standard TCP</td>
</tr>
<tr>
<td>23</td>
<td>Secure Tunnel</td>
</tr>
<tr>
<td>24</td>
<td>Integrated SSH</td>
</tr>
</tbody>
</table>
The Standard TCP channel type represents a generic TCP connection. The Secure Tunnel and Integrated SSH channel types are described in Section 2.3.

Values of the "enum<idhash>" parameter represent secure hash algorithms in the "VNC Hash Algorithms" registry created by Section 4.4 of this document. The initial values include:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MD5</td>
</tr>
<tr>
<td>2</td>
<td>SHA1</td>
</tr>
<tr>
<td>4</td>
<td>SHA256</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>BPP</th>
<th>Depth</th>
<th>TC</th>
<th>RM</th>
<th>GM</th>
<th>BM</th>
<th>RS</th>
<th>GS</th>
<th>BS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Black and White</td>
<td>8</td>
<td>3</td>
<td>t</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>Greyscale</td>
<td>8</td>
<td>6</td>
<td>t</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>8 Colors</td>
<td>8</td>
<td>3</td>
<td>t</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>64 Colors</td>
<td>8</td>
<td>3</td>
<td>t</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>256 Colors</td>
<td>8</td>
<td>8</td>
<td>t</td>
<td>7</td>
<td>7</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>16-bit Color</td>
<td>16</td>
<td>16</td>
<td>t</td>
<td>31</td>
<td>63</td>
<td>31</td>
<td>11</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>24-bit Color</td>
<td>32</td>
<td>24</td>
<td>t</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>16</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>30-bit Color</td>
<td>32</td>
<td>30</td>
<td>t</td>
<td>1023</td>
<td>1023</td>
<td>1023</td>
<td>0</td>
<td>10</td>
<td>20</td>
</tr>
</tbody>
</table>

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

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

o  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&

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 may 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. 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. Canceling the dialog or ending the session SHOULD terminate the application.
2.2.1. Error Handling

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

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, 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 MUST alert the user. The alert MUST provide the remote system’s identifying information including the hash value and allow the user to terminate the connection. The alert MAY allow the user to accept the key and continue establishing the connection.

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 MAY 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 SHOULD treat the value as a request to use "Integrated SSH" as the "ChannelType". However this value SHOULD NOT be supplied for the "SecurityType" parameter as the channel must be 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 MUST alert the user. The alert MUST provide the remote system’s identifying information and allow the user to terminate the connection. The alert MAY allow the user to accept the key and continue establishing the connection.

When providing identifying information of a host identified by an X509 certificate [RFC5280], the certificate subject, issuer, validity period, and certificate hash MUST be included. The VNC client MAY verify the validity of the certificate. If the validity of a certificate is not determined, the console application MUST include 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 are 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 SHOULD treat the value as a request to use "Secure Tunnel" as the "ChannelType". However this value SHOULD NOT be supplied for the "SecurityType" parameter 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 may be included in a URI. If those credentials are not kept secure, an attacker may gain access to any systems using those credentials. Host addresses and connection parameters may 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 connected to. If a user connects to an attacker’s system, user data, including credentials, may be exposed.

Note that the RFB protocol itself does 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 should 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 MAY wish 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 MAY verify its caller thorough platform-specific means.

VNC clients SHOULD NOT 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 OS.

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

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

Applications which 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 which 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 MAY 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 MUST alert the user. The alert MUST provide the remote system’s identifying information and allow the user to terminate the connection. The alert MAY allow the user to accept the information and continue establishing the connection.

When providing identifying information of a host identified by an X509 certificate [RFC5280], the certificate subject, issuer, validity period, and certificate hash MUST be included. The VNC client MAY verify the certificate validity. If the validity of a certificate is not determined, the console application MUST 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, MUST include a hash of the key.

3.4. Connection Database Integrity

A VNC client application and/or launching application MAY maintain a connection database containing remote host information, credentials, and/or connection parameters. Applications MUST NOT store credentials unless the credentials are stored in an encrypted format with a decryption process requiring user-supplied or device-specific data. If supported, an application SHOULD have a setting disabling storage of credentials.

If available, the VNC client connection database SHOULD 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 MUST 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 Connection Channel Types", "VNC 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 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. IANA is asked to update the "Secure Tunnel" and "Integrated SSH" security types to refer to this document.

Future RFB security types will be supported in "vnc" URIs. Any new security mechanism integrated with a VNC client which alters the process by which a connection is established should also be reflected in the registry to avoid overlapping numeric assignments.

This applies even if the mechanism does not involve a change to the VNC server implementation or RFB protocol itself. For example, the "Secure Tunnel" security type does not require RFB modification, but indicates to the client that it must establish a TLS tunnel prior to initiating RFB communications. Without inclusion in the registry, a URI-launched client will be unable to distinguish future client-initiated and protocol-based security mechanisms.

4.3. VNC Connection Channel Types

IANA is asked to create a "VNC Connection Channel Types" registry within the "Remote Framebuffer (RFB)" 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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Standard TCP</td>
<td>(this document)</td>
</tr>
<tr>
<td>23</td>
<td>Secure Tunnel</td>
<td>(this document)</td>
</tr>
<tr>
<td>24</td>
<td>Integrated SSH</td>
<td>(this document)</td>
</tr>
</tbody>
</table>

Future assignments to this registry should be made through the "Expert Review" process described in [RFC5226]. Experts reviewing
VNC 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.

### 4.4. VNC ID Hash Algorithms

IANA is asked to create a "VNC ID Hash Algorithms" registry within the "Remote Framebuffer (RFB)" 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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MD5</td>
<td>(this document)</td>
</tr>
<tr>
<td>2</td>
<td>SHA1</td>
<td>(this document)</td>
</tr>
<tr>
<td>4</td>
<td>SHA256</td>
<td>(this document)</td>
</tr>
</tbody>
</table>

Future assignments to this registry should be made through the "Expert Review" process described in [RFC5226]. Experts reviewing VNC 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.

### 4.5. VNC URI Parameters

IANA is asked to create a "VNC URI Parameters" registry within the "Remote Framebuffer (RFB)" group.

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConnectionName</td>
<td>Name of connection profile</td>
<td>(this document)</td>
</tr>
<tr>
<td>VncUsername</td>
<td>VNC server username</td>
<td>(this document)</td>
</tr>
<tr>
<td>VncPassword</td>
<td>VNC server password</td>
<td>(this document)</td>
</tr>
<tr>
<td>SecurityType</td>
<td>VNC security type used</td>
<td>(this document)</td>
</tr>
<tr>
<td>ChannelType</td>
<td>Connection channel type</td>
<td>(this document)</td>
</tr>
</tbody>
</table>
### 4. VNC URI Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>SshHost</td>
<td>SSH server hostname or IP</td>
<td>(this document)</td>
</tr>
<tr>
<td>SshPort</td>
<td>SSH server port</td>
<td>(this document)</td>
</tr>
<tr>
<td>SshUsername</td>
<td>SSH username</td>
<td>(this document)</td>
</tr>
<tr>
<td>SshPassword</td>
<td>SSH password</td>
<td>(this document)</td>
</tr>
<tr>
<td>IdHashAlgorithm</td>
<td>Hash algorithm used with &quot;IdHash&quot; parameter</td>
<td>(this document)</td>
</tr>
<tr>
<td>IdHash</td>
<td>Expected hash of remote public key or certificate</td>
<td>(this document)</td>
</tr>
<tr>
<td>ColorLevel</td>
<td>Client color depth/mode</td>
<td>(this document)</td>
</tr>
<tr>
<td>ViewOnly</td>
<td>Client is view only</td>
<td>(this document)</td>
</tr>
<tr>
<td>SaveConnection</td>
<td>Store connection info</td>
<td>(this document)</td>
</tr>
</tbody>
</table>

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


5.2. Informative References


6. Acknowledgments

Dominic Parkes and the staff of RealVNC Ltd. graciously reviewed this document and provided constructive comments.
RFB and VNC are registered trademarks of RealVNC Ltd. in the U.S. and in other countries.

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: IETF Applications Area Directors <app-ads@tools.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