

[MS-RDPELE]:

Remote Desktop Protocol: Licensing Extension

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation (“this documentation”) for protocols, file formats, data portability, computer languages, and standards support. Additionally, overview documents cover inter-protocol relationships and interactions.
- **Copyrights.** This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you can make copies of it in order to develop implementations of the technologies that are described in this documentation and can distribute portions of it in your implementations that use these technologies or in your documentation as necessary to properly document the implementation. You can also distribute in your implementation, with or without modification, any schemas, IDLs, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications documentation.
- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.
- **Patents.** Microsoft has patents that might cover your implementations of the technologies described in the Open Specifications documentation. Neither this notice nor Microsoft's delivery of this documentation grants any licenses under those patents or any other Microsoft patents. However, a given Open Specifications document might be covered by the Microsoft [Open Specifications Promise](#) or the [Microsoft Community Promise](#). If you would prefer a written license, or if the technologies described in this documentation are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **License Programs.** To see all of the protocols in scope under a specific license program and the associated patents, visit the [Patent Map](#).
- **Trademarks.** The names of companies and products contained in this documentation might be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights. For a list of Microsoft trademarks, visit www.microsoft.com/trademarks.
- **Fictitious Names.** The example companies, organizations, products, domain names, email addresses, logos, people, places, and events that are depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

Reservation of Rights. All other rights are reserved, and this notice does not grant any rights other than as specifically described above, whether by implication, estoppel, or otherwise.

Tools. The Open Specifications documentation does not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments, you are free to take advantage of them. Certain Open Specifications documents are intended for use in conjunction with publicly available standards specifications and network programming art and, as such, assume that the reader either is familiar with the aforementioned material or has immediate access to it.

Support. For questions and support, please contact dochelp@microsoft.com.

Revision Summary

Date	Revision History	Revision Class	Comments
7/20/2007	0.1	Major	MCPPE Milestone 5 Initial Availability
9/28/2007	0.2	Minor	Clarified the meaning of the technical content.
10/23/2007	0.3	Minor	Clarified the meaning of the technical content.
11/30/2007	0.4	Minor	Clarified the meaning of the technical content.
1/25/2008	0.4.1	Editorial	Changed language and formatting in the technical content.
3/14/2008	0.5	Minor	Clarified the meaning of the technical content.
5/16/2008	0.5.1	Editorial	Changed language and formatting in the technical content.
6/20/2008	1.0	Major	Updated and revised the technical content.
7/25/2008	2.0	Major	Updated and revised the technical content.
8/29/2008	2.1	Minor	Clarified the meaning of the technical content.
10/24/2008	2.2	Minor	Clarified the meaning of the technical content.
12/5/2008	3.0	Major	Updated and revised the technical content.
1/16/2009	3.0.1	Editorial	Changed language and formatting in the technical content.
2/27/2009	4.0	Major	Updated and revised the technical content.
4/10/2009	4.1	Minor	Clarified the meaning of the technical content.
5/22/2009	4.1.1	Editorial	Changed language and formatting in the technical content.
7/2/2009	4.1.2	Editorial	Changed language and formatting in the technical content.
8/14/2009	4.2	Minor	Clarified the meaning of the technical content.
9/25/2009	4.3	Minor	Clarified the meaning of the technical content.
11/6/2009	4.3.1	Editorial	Changed language and formatting in the technical content.
12/18/2009	5.0	Major	Updated and revised the technical content.
1/29/2010	5.1	Minor	Clarified the meaning of the technical content.
3/12/2010	6.0	Major	Updated and revised the technical content.
4/23/2010	7.0	Major	Updated and revised the technical content.
6/4/2010	8.0	Major	Updated and revised the technical content.
7/16/2010	8.1	Minor	Clarified the meaning of the technical content.
8/27/2010	9.0	Major	Updated and revised the technical content.
10/8/2010	10.0	Major	Updated and revised the technical content.
11/19/2010	10.0	None	No changes to the meaning, language, or formatting of the technical content.
1/7/2011	10.0	None	No changes to the meaning, language, or formatting of the

Date	Revision History	Revision Class	Comments
			technical content.
2/11/2011	10.0	None	No changes to the meaning, language, or formatting of the technical content.
3/25/2011	10.0	None	No changes to the meaning, language, or formatting of the technical content.
5/6/2011	10.0	None	No changes to the meaning, language, or formatting of the technical content.
6/17/2011	10.1	Minor	Clarified the meaning of the technical content.
9/23/2011	10.1	None	No changes to the meaning, language, or formatting of the technical content.
12/16/2011	11.0	Major	Updated and revised the technical content.
3/30/2012	11.0	None	No changes to the meaning, language, or formatting of the technical content.
7/12/2012	11.1	Minor	Clarified the meaning of the technical content.
10/25/2012	11.2	Minor	Clarified the meaning of the technical content.
1/31/2013	11.2	None	No changes to the meaning, language, or formatting of the technical content.
8/8/2013	11.2	None	No changes to the meaning, language, or formatting of the technical content.
11/14/2013	11.2	None	No changes to the meaning, language, or formatting of the technical content.
2/13/2014	11.2	None	No changes to the meaning, language, or formatting of the technical content.
5/15/2014	11.2	None	No changes to the meaning, language, or formatting of the technical content.
6/30/2015	12.0	Major	Significantly changed the technical content.
10/16/2015	12.0	None	No changes to the meaning, language, or formatting of the technical content.
7/14/2016	12.1	Minor	Clarified the meaning of the technical content.
6/1/2017	12.1	None	No changes to the meaning, language, or formatting of the technical content.
9/15/2017	13.0	Major	Significantly changed the technical content.

Table of Contents

1	Introduction	7
1.1	Glossary	7
1.2	References	9
1.2.1	Normative References	9
1.2.2	Informative References	10
1.3	Overview	10
1.3.1	Licensing Architecture	10
1.3.2	X.509 Certificate Chains	12
1.3.3	Licensing PDU Flows.....	12
1.3.3.1	New License Flow.....	14
1.3.3.2	Upgrade License Flow	14
1.4	Relationship to Other Protocols	15
1.5	Prerequisites/Preconditions	15
1.6	Applicability Statement	15
1.7	Versioning and Capability Negotiation	15
1.8	Vendor-Extensible Fields	15
1.9	Standards Assignments.....	15
2	Messages.....	16
2.1	Transport	16
2.2	Message Syntax	16
2.2.1	Common Data Structures	16
2.2.1.1	Security Headers	16
2.2.1.1.1	Basic (TS_SECURITY_HEADER)	16
2.2.1.1.2	Non-FIPS (TS_SECURITY_HEADER1)	16
2.2.1.1.3	FIPS (TS_SECURITY_HEADER2)	16
2.2.1.2	Licensing Preamble (LICENSE_PREAMBLE)	16
2.2.1.3	Licensing Binary BLOB (LICENSE_BINARY_BLOB).....	16
2.2.1.4	Server Certificate (SERVER_CERTIFICATE)	16
2.2.1.4.1	Server Proprietary Certificate (PROPRIETARYSERVERCERTIFICATE)	16
2.2.1.4.2	X.509 Certificate Chain (X509_CERTIFICATE_CHAIN)	16
2.2.1.4.2.1	CertBlob (CERT_BLOB)	17
2.2.1.4.3	Proprietary Certificate (PROPRIETARYSERVERCERTIFICATE)	17
2.2.2	Licensing PDU (TS_LICENSING_PDU)	17
2.2.2.1	Server License Request (SERVER_LICENSE_REQUEST)	20
2.2.2.1.1	Product Information (PRODUCT_INFO)	21
2.2.2.1.2	Scope List (SCOPE_LIST)	22
2.2.2.1.2.1	Scope (SCOPE)	22
2.2.2.2	Client New License Request (CLIENT_NEW_LICENSE_REQUEST)	22
2.2.2.3	Client License Information (CLIENT_LICENSE_INFO)	24
2.2.2.3.1	Client Hardware Identification (CLIENT_HARDWARE_ID)	25
2.2.2.4	Server Platform Challenge (SERVER_PLATFORM_CHALLENGE)	26
2.2.2.5	Client Platform Challenge Response (CLIENT_PLATFORM_CHALLENGE_RESPONSE)	26
2.2.2.5.1	Platform Challenge Response Data (PLATFORM_CHALLENGE_RESPONSE_DATA)	27
2.2.2.6	Server Upgrade License (SERVER_UPGRADE_LICENSE)	28
2.2.2.6.1	New License Information (NEW_LICENSE_INFO)	29
2.2.2.7	Server New License (SERVER_NEW_LICENSE)	30
2.2.2.7.1	License Error Message (LICENSE_ERROR_MESSAGE)	30
3	Protocol Details.....	31
3.1	Common Details	31
3.1.1	Abstract Data Model.....	31
3.1.2	Timers	31

3.1.3	Initialization	31
3.1.4	Higher-Layer Triggered Events	31
3.1.5	Message Processing Events and Sequencing Rules	31
3.1.5.1	Message Integrity Checking.....	31
3.1.5.2	Sending License Error Messages	31
3.1.5.3	Processing License Error Messages	32
3.1.5.3.1	Client State Transition.....	32
3.1.5.3.2	Server State Transition	32
3.1.6	Timer Events.....	33
3.1.7	Other Local Events.....	33
3.2	Server Details.....	33
3.2.1	Abstract Data Model.....	33
3.2.1.1	Server Random.....	33
3.2.1.2	Product Information	33
3.2.1.3	Server Certificate	34
3.2.1.4	Key Exchange List.....	34
3.2.1.5	Scope List	34
3.2.1.6	Platform Challenge	34
3.2.1.7	License	34
3.2.1.8	ClientUserName	34
3.2.1.9	ClientMachineName.....	34
3.2.1.10	Encryption Keys.....	35
3.2.1.11	Server Licensing States	35
3.2.2	Timers	35
3.2.3	Initialization	35
3.2.4	Higher-Layer Triggered Events	35
3.2.5	Message Processing Events and Sequencing Rules	35
3.2.5.1	Sending Server License Request PDUs	35
3.2.5.2	Processing Client New License Requests.....	35
3.2.5.3	Processing Client License Information	36
3.2.5.4	Sending Server Platform Challenges	37
3.2.5.5	Processing Client Platform Challenge Responses	37
3.2.5.6	Sending Server Upgrade Licenses	38
3.2.5.7	Sending Server New Licenses	38
3.2.5.8	Handling Out-of-Sequence or Unrecognized Messages	38
3.2.5.9	Handling Invalid MACs	39
3.2.6	Timer Events.....	39
3.2.7	Other Local Events.....	39
3.3	Client Details	39
3.3.1	Abstract Data Model.....	39
3.3.1.1	Platform ID	39
3.3.1.2	Client Random.....	39
3.3.1.3	Preferred Key Exchange Algorithm ID	39
3.3.1.4	Client User Name	39
3.3.1.5	Client Machine Name.....	39
3.3.1.6	Encrypted Premaster Secret	39
3.3.1.7	License	40
3.3.1.8	License Store.....	40
3.3.1.9	Client Hardware Identification.....	40
3.3.1.10	Encryption Keys.....	40
3.3.1.11	Client Licensing States.....	40
3.3.2	Timers	41
3.3.2.1	Client Packet Wait Timer	41
3.3.3	Initialization.....	41
3.3.4	Higher-Layer Triggered Events	41
3.3.5	Message Processing Events and Sequencing Rules	41
3.3.5.1	Processing Server License Requests	41
3.3.5.2	Sending Client New License Requests	41

3.3.5.3	Sending Client License Information	41
3.3.5.4	Processing Server Platform Challenges	42
3.3.5.5	Sending Client Platform Challenge Responses	42
3.3.5.6	Processing Server Upgrade Licenses	42
3.3.5.7	Processing Server New Licenses.....	42
3.3.5.8	Handling Out-of-Sequence or Unrecognized Messages	42
3.3.5.9	Handling Invalid MACs	43
3.3.6	Timer Events.....	43
3.3.7	Other Local Events.....	43
4	Protocol Examples	44
4.1	SERVER LICENSE REQUEST	44
4.2	CLIENT NEW LICENSE REQUEST	49
4.3	CLIENT LICENSE INFO	50
4.4	SERVER PLATFORM CHALLENGE.....	55
4.5	CLIENT PLATFORM CHALLENGE RESPONSE	56
4.6	SERVER NEW LICENSE.....	57
4.7	SERVER UPGRADE LICENSE.....	64
5	Security	72
5.1	Security Considerations for Implementers	72
5.1.1	X.509 Certificate.....	72
5.1.2	Client and Server Random Values and Premaster Secrets	72
5.1.2.1	Encrypting the Premaster Secret.....	73
5.1.2.2	Decrypting the Premaster Secret.....	73
5.1.3	Generating the Licensing Encryption and MAC Salt Keys	73
5.1.4	Encrypting Licensing Session Data	74
5.1.5	Decrypting Licensing Session Data.....	74
5.1.6	MAC Generation	74
5.2	Index of Security Parameters	74
6	Appendix A: Product Behavior	75
7	Change Tracking.....	78
8	Index.....	79

1 Introduction

The Remote Desktop Protocol: Licensing Extension expands on the licensing protocol sequence specified in [\[MS-RDPBCGR\]](#).

Sections 1.5, 1.8, 1.9, 2, and 3 of this specification are normative. All other sections and examples in this specification are informative.

1.1 Glossary

This document uses the following terms:

Active Directory: A general-purpose network directory service. **Active Directory** also refers to the Windows implementation of a directory service. **Active Directory** stores information about a variety of objects in the network. User accounts, computer accounts, groups, and all related credential information used by the Windows implementation of Kerberos are stored in **Active Directory**. **Active Directory** is either deployed as Active Directory Domain Services (AD DS) or Active Directory Lightweight Directory Services (AD LDS). [\[MS-ADTS\]](#) describes both forms. For more information, see [\[MS-AUTHSOD\]](#) section 1.1.1.5.2, Lightweight Directory Access Protocol (LDAP) versions 2 and 3, Kerberos, and DNS.

American National Standards Institute (ANSI) character set: A character set defined by a code page approved by the American National Standards Institute (ANSI). The term "ANSI" as used to signify Windows code pages is a historical reference and a misnomer that persists in the Windows community. The source of this misnomer stems from the fact that the Windows code page 1252 was originally based on an ANSI draft, which became International Organization for Standardization (ISO) Standard 8859-1 [\[ISO/IEC-8859-1\]](#). In Windows, the ANSI character set can be any of the following code pages: 1252, 1250, 1251, 1253, 1254, 1255, 1256, 1257, 1258, 874, 932, 936, 949, or 950. For example, "ANSI application" is usually a reference to a non-Unicode or code-page-based application. Therefore, "ANSI character set" is often misused to refer to one of the character sets defined by a Windows code page that can be used as an active system code page; for example, character sets defined by code page 1252 or character sets defined by code page 950. Windows is now based on Unicode, so the use of ANSI character sets is strongly discouraged unless they are used to interoperate with legacy applications or legacy data.

clearing house: A Microsoft central authority for activating a **license server** and registering **client access licenses (CALs)**.

client: A computer on which the remote procedure call (RPC) client is executing.

client access license (CAL): A license required by a **client** user or device for accessing a **terminal server** configured in Application Server mode.

client license: See **client access license (CAL)**.

grace period: The duration of time during which a **terminal server** allows **clients** to connect without requiring a CAL. The grace period ends either when the duration is complete or when the **terminal server** receives the first **permanent license** from the **license server**.

license encryption key: A shared symmetric key generated by both the **server** and **client** that is used to encrypt licensing message data.

license server: A **server** that issues **CALs**.

license server certificate: An X.509 certificate used for signing **CALs**.

license store: A **client**-side database that stores **CALs** issued by a **terminal server**.

MD5 digest: A 128-bit message hash value generated as output by the MD5 Message-Digest algorithm. See [\[RFC1321\]](#).

Message Authentication Code (MAC): A message authenticator computed through the use of a symmetric key. A **MAC** algorithm accepts a secret key and a data buffer, and outputs a **MAC**. The data and **MAC** can then be sent to another party, which can verify the integrity and authenticity of the data by using the same secret key and the same **MAC** algorithm.

object identifier (OID): In the context of a directory service, a number identifying an object class or attribute. Object identifiers are issued by the ITU and form a hierarchy. An OID is represented as a dotted decimal string (for example, "1.2.3.4"). For more information on OIDs, see [\[X660\]](#) and [\[RFC3280\]](#) Appendix A. OIDs are used to uniquely identify certificate templates available to the certification authority (CA). Within a certificate, OIDs are used to identify standard extensions, as described in [\[RFC3280\]](#) section 4.2.1.x, as well as non-standard extensions.

permanent license: A CAL issued to authenticated clients.

personal terminal server: In general context, refers to a **client** SKU target machine that hosts remote desktop sessions. From a terminal service licensing perspective, the behavior of a **personal terminal server** is similar to that of a **terminal server** in **remote administration mode**. Thus any behavioral reference to a **personal terminal server** in this document essentially implies that the particular behavior is valid for a **terminal server** in **remote administration mode** as well. The term **personal terminal server** is therefore used to encompass all connections where either the end point is a **client** SKU operating system or is a **terminal server** running in **remote administration mode**.

premaster secret: A 48-byte random number used in **license encryption key** generation.

RC4: A variable key-length symmetric encryption algorithm. For more information, see [\[SCHNEIER\]](#) section 17.1.

remote administration mode: A **terminal server** can function in **remote administration mode** if either the terminal services role is not installed on the machine or the **client** used to invoke the session has enabled the /admin switch. The administrator can log in to the **terminal server** in the **remote administration mode** by using the following command from any **Remote Desktop client** (with Terminal Services Client version 6.0 or 6.1). "mstsc /admin <remote machine name>"

Remote Desktop client: A device that connects to a **terminal server** and renders the user interface through which a user interacts with a remote session.

Remote Desktop Protocol (RDP): A multi-channel protocol that allows a user to connect to a computer running Microsoft Terminal Services (TS). RDP enables the exchange of client and server settings and also enables negotiation of common settings to use for the duration of the connection, so that input, graphics, and other data can be exchanged and processed between client and server.

server: A computer on which the remote procedure call (RPC) server is executing.

session encryption key: A shared key used for confidential exchange of data between the **client** and the **server**.

SHA-1 hash: A hashing algorithm as specified in [\[FIPS180-2\]](#) that was developed by the National Institute of Standards and Technology (NIST) and the National Security Agency (NSA).

temporary license: A type of CAL issued by a **terminal server** to a **client** in situations in which a **permanent license** is not available.

terminal server: The server to which a client initiates a remote desktop connection. The **server** hosts Remote Desktop sessions and enables interaction with each of these sessions on a connected **client** device.

terminal server certificate: A certificate that should be used to authenticate a **terminal server**.

Unicode string: A Unicode 8-bit string is an ordered sequence of 8-bit units, a Unicode 16-bit string is an ordered sequence of 16-bit code units, and a Unicode 32-bit string is an ordered sequence of 32-bit code units. In some cases, it could be acceptable not to terminate with a terminating null character. Unless otherwise specified, all **Unicode strings** follow the UTF-16LE encoding scheme with no Byte Order Mark (BOM).

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as defined in [\[RFC2119\]](#). All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

1.2 References

Links to a document in the Microsoft Open Specifications library point to the correct section in the most recently published version of the referenced document. However, because individual documents in the library are not updated at the same time, the section numbers in the documents may not match. You can confirm the correct section numbering by checking the [Errata](#).

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information.

[ISO/IEC-8859-1] International Organization for Standardization, "Information Technology -- 8-Bit Single-Byte Coded Graphic Character Sets -- Part 1: Latin Alphabet No. 1", ISO/IEC 8859-1, 1998, http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=28245

Note There is a charge to download the specification.

[MS-RDPBCGR] Microsoft Corporation, "[Remote Desktop Protocol: Basic Connectivity and Graphics Remoting](#)".

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

[RFC3280] Housley, R., Polk, W., Ford, W., and Solo, D., "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 3280, April 2002, <http://www.ietf.org/rfc/rfc3280.txt>

[T123] ITU-T, "Network-Specific Data Protocol Stacks for Multimedia Conferencing", Recommendation T.123, May 1999, <http://www.itu.int/rec/T-REC-T.123/en>

Note There is a charge to download the specification.

[T125] ITU-T, "Multipoint Communication Service Protocol Specification", Recommendation T.125, February 1998, <http://www.itu.int/rec/T-REC-T.125-199802-I/en>

Note There is a charge to download the specification.

[X224] ITU-T, "Information technology - Open Systems Interconnection - Protocol for Providing the Connection-Mode Transport Service", Recommendation X.224, November 1995, <http://www.itu.int/rec/T-REC-X.224-199511-I/en>

Note There is a charge to download the specification.

1.2.2 Informative References

[MS-EERR] Microsoft Corporation, "[ExtendedError Remote Data Structure](#)".

[MSDN-CAI] Microsoft Corporation, "CRYPT_ALGORITHM_IDENTIFIER structure", <http://msdn.microsoft.com/En-US/library/aa381133.aspx>

[MSDN-OSVER] Microsoft Corporation, "Operating System Version", [http://msdn.microsoft.com/en-us/library/ms724832\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/ms724832(VS.85).aspx)

[MSDN-RC4] Microsoft Corporation, "MSDN Security Glossary", <http://msdn.microsoft.com/en-us/library/ms721604.aspx>

[SCHNEIER] Schneier, B., "Applied Cryptography, Second Edition", John Wiley and Sons, 1996, ISBN: 0471117099, <http://www.wiley.com/WileyCDA/WileyTitle/productCd-0471117099.html>

1.3 Overview

The Remote Desktop Protocol: Licensing Extension is designed to allow authorized **remote desktop clients** or users to connect to a **terminal server** (A reference to terminal server in this document generally implies a terminal server in app-server mode). It involves communication between a Remote Desktop client, a terminal server, and a **license server**. The terminal server can be configured to function in per-device or per-user license mode. **Client access licenses (CALs)** are installed on a license server, so that when a terminal server requests a license on a client's behalf, the license server issues a license out of its available pool of licenses.

The Licensing Extension also provides a mechanism for remote desktop clients to send their client access licenses (CALs) to **personal terminal servers**. The presence of these CALs in such connections is not designed to control access to the target machines but can be used to turn on or off specific RDP features reserved for licensed clients. The communication involved here is limited to communication between the Remote Desktop client and the personal terminal server. No license server is involved in any part of the communication.

1.3.1 Licensing Architecture

The Remote Desktop Protocol: Licensing Extension involves the following components:

- **Remote desktop client:** Connects to a **terminal server** and renders the user interface through which a user interacts with a remote session.
- **Terminal server:** Hosts remote desktop sessions and enables interaction with each of these sessions on a connected client device. A reference to terminal server generally refers to a terminal server in app-server mode.
- **Personal terminal server:** Hosts remote desktop session where the target operating system is either a client SKU or a terminal server in **remote administration mode**.
- **License server:** Issues licenses to users or devices using remote desktop sessions.
- **Clearing house:** Activates license servers and supplies CALs to license servers.
- **Active Directory:** Stores licenses issued to users.
- **License Manager:** Administers license servers.

The following diagram illustrates the relationship and interaction among these components in a typical terminal server deployment.

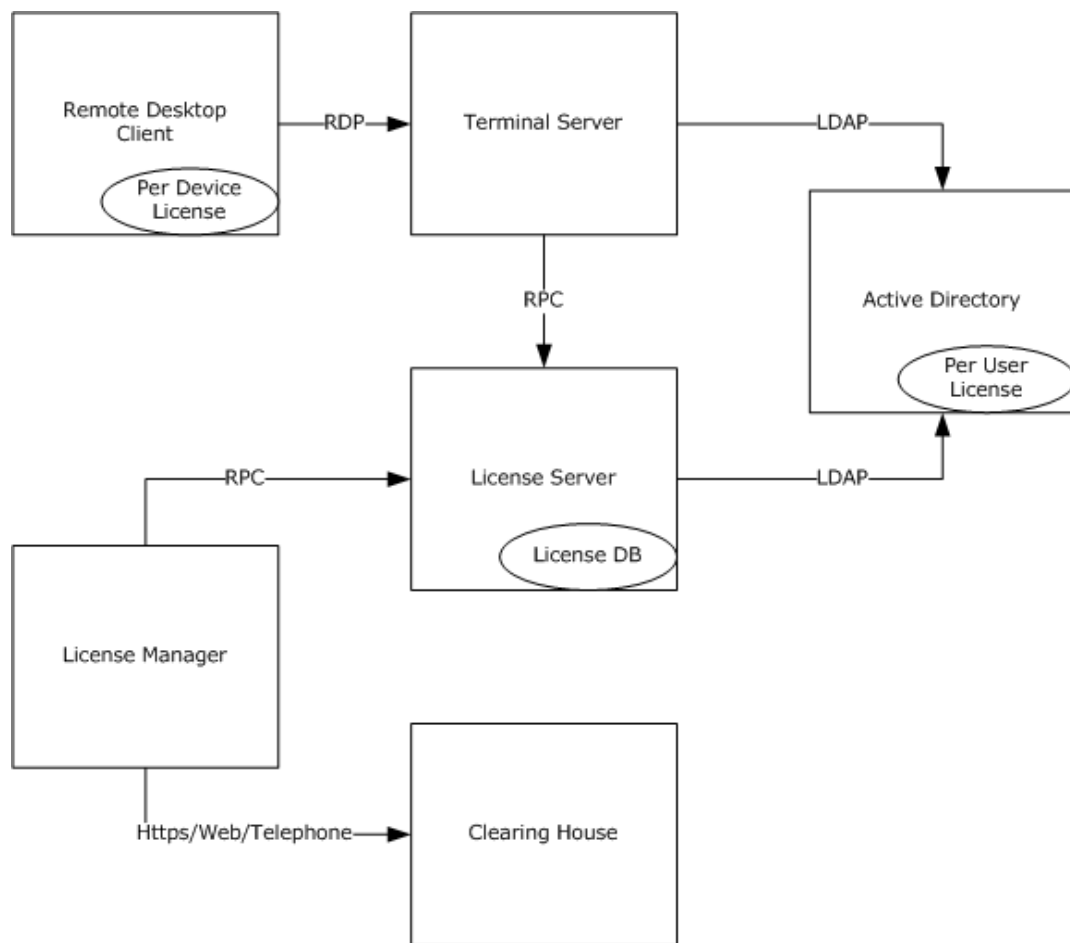


Figure 1: Licensing architecture components for terminal server licensing

The Remote Desktop Protocol: Licensing Extension facilitates the exchange of licensing information between the Remote Desktop client and the terminal server and is restricted between these two components only. The interaction among the remaining components (for example, between the terminal server and the license server) is not a function of the Remote Desktop Protocol: Licensing Extension.

The license server manages the terminal server CALs. It keeps track of issued and expired per-device terminal server CALs in the license database. The per-user terminal server CALs are stored in Active Directory. The terminal server communicates with the license server, using RPC to accomplish the following tasks:

- Issue a new terminal server CAL.
- Upgrade an older version terminal server CAL.

The terminal server interacts with Active Directory to retrieve information on the per-user terminal server CAL when the terminal server is configured in per-user license mode. The license server has to be registered with the Microsoft clearing house before it starts issuing terminal server CALs to the Remote desktop clients. The License Manager is the GUI application for managing the license server. The License Manager provides an interface to the administrator to register the license server with the clearing house. The administrator can use one of two methods to register the license server with the clearing house:

- **HTTPS:** The License Manager contacts the clearing house over HTTPS and registers the license server and terminal server CAL key-packs.
- **Telephone/web:** The administrator gets the license server and terminal server CAL key-pack registration information manually from the telephone or web and enters the registration information in the interface provided by the License Manager.

The Remote Desktop Protocol: The personal terminal server and the Remote Desktop client do not exchange any licensing information.

For more information about license PDU flows for personal terminal servers, see section [1.3.3](#).

1.3.2 X.509 Certificate Chains

A **license server** issues X.509 certificate chains (see [\[RFC3280\]](#)) to terminal servers and **Remote Desktop clients**. A certificate chain is a sequence of certificates. The chain usually starts with a leaf certificate and terminates at a root certificate. Each certificate is signed by the subject of the subsequent certificate in the chain. The root certificate is self-signed. For the structure of X.509 certificate chains used in the Remote Desktop Protocol: Licensing Extension, see section [2.2.1.4.2](#).

The certificate encoding used is ASN.1 DER, as specified in [\[RFC3280\]](#) section 4.1.

1.3.3 Licensing PDU Flows

A target machine (**terminal server** or **personal terminal server**) initiates the licensing protocol data unit (PDU) exchange by sending a [Server License Request](#) message on receipt of the Client Info PDU (see [\[MS-RDPBCGR\]](#) section 2.2.1.11 and [\[MS-RDPBCGR\]](#) section 3.3.5.3.11).

When a **Remote Desktop client** connects to a target machine, either the client has a license or it does not.

If the client is connecting to a terminal server and the client does not have a license, the terminal server tries to obtain a license (See [New License Flow \(section 1.3.3.1\)](#).) If the client has a license, the terminal server validates the version and expiry date. If the license is valid, access is allowed. If the license is expired, temporary, or a lower version than the operating system version of the terminal server, the license is upgraded. For the steps to upgrade the license, see [Upgrade License Flow \(section 1.3.3.2\)](#).

If the target machine is a personal terminal server, whether the client sends the license or not, the server always sends a license error message with the error code STATUS_VALID_CLIENT and the state transition code ST_NO_TRANSITION. Also, in the case that the client sends a license, the server does not validate it. The licensing protocol is complete at this point.

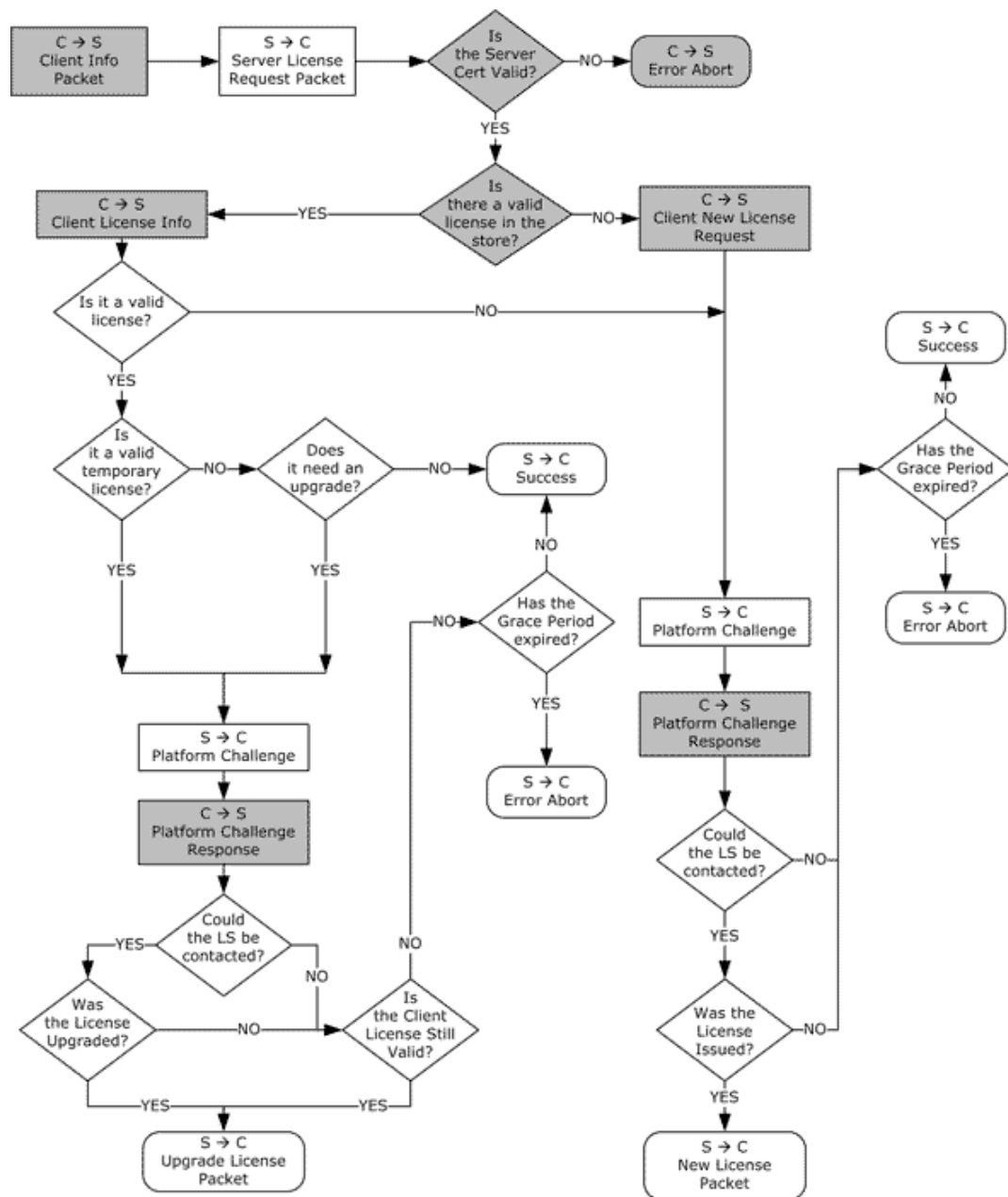


Figure 2: Licensing PDU flows in Terminal Server

This flow chart describes the logic for the following cases:

- A license is issued through the terminal server. The server issues a CAL to the client when the client does not have a license in its **license store**.
- A license is upgraded. A CAL is upgraded when it is a **temporary license** or a **permanent license** that is going to expire in seven days. A valid license has to be upgraded if the license is meant for an older version of the terminal server or if it has expired.
- An error condition occurs.

1.3.3.1 New License Flow

When the **Remote Desktop client** does not have a license in its **license store**, the message flow is as shown in the following diagram.

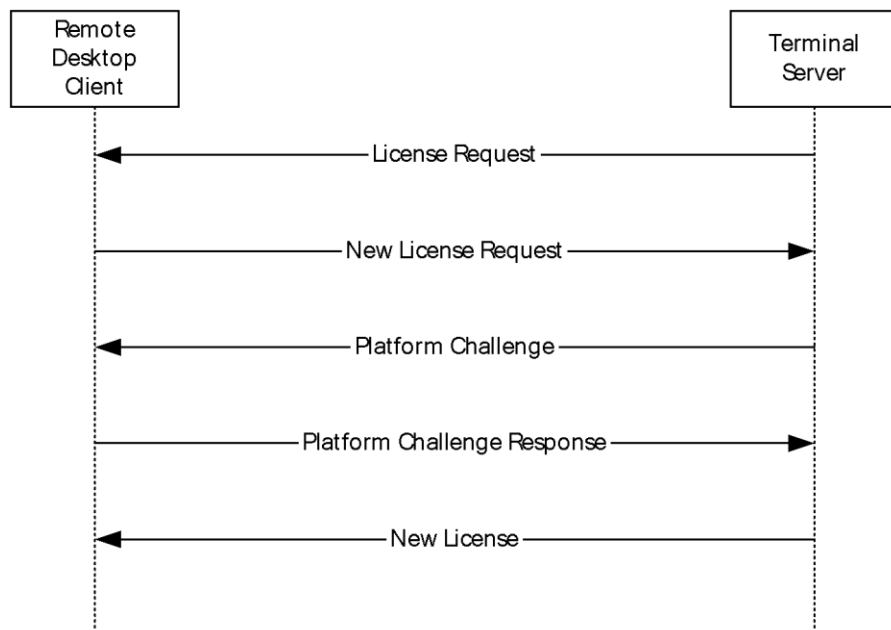


Figure 3: Remote Desktop client new license flow

1.3.3.2 Upgrade License Flow

When the **Remote Desktop client** has a license in its **license store**, the message flow is as shown in the following diagram.

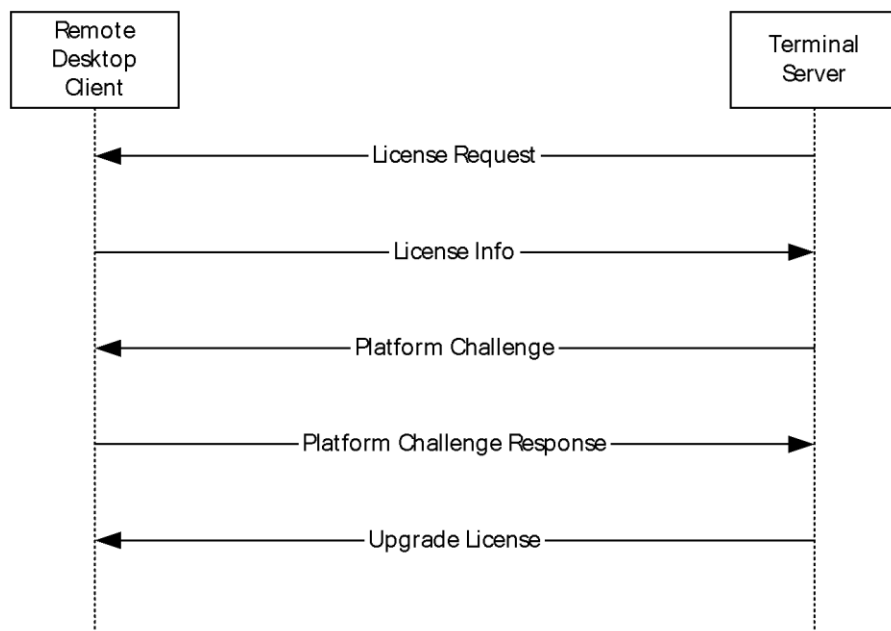


Figure 4: Remote Desktop client upgrade license flow

1.4 Relationship to Other Protocols

The Remote Desktop Protocol: Licensing Extension extends the Remote Desktop Protocol: Basic Connectivity and Graphics Remoting (as specified in [\[MS-RDPBCGR\]](#)) by adding licensing capabilities.

The licensing protocol sequence is started by the server during the **Remote Desktop Protocol (RDP)** standard connection sequence after receiving the Client Info PDU (see sections 1.3.1.1 and 2.2.11). If exchange of licensing information is required, the server sends a [Server License Request \(section 2.2.2.1\)](#) to the client at this point. Otherwise, the RDP standard connection sequence continues (see section 1.3.1.1).

1.5 Prerequisites/Preconditions

The Remote Desktop Protocol: Licensing Extension assumes that the system already has an IP address and is thus capable of communicating on the network. It also assumes that the initiator (or **client**) has already obtained the IP address of the **server**, that the server has registered a port, and that the server is actively listening for client connections on that port.

All multiple-byte fields within a message are assumed to contain data in little-endian byte ordering, unless otherwise specified.

1.6 Applicability Statement

The Remote Desktop Protocol: Licensing Extension applies whenever a **Remote Desktop client** attempts to connect to a **terminal server** in Application Server mode and exchange of licensing information is required. The licensing protocol details provided in this document allow a server to authorize a client connection by issuing and verifying **CALs**.[<1>](#)

1.7 Versioning and Capability Negotiation

Only one version of the Remote Desktop Protocol: Licensing Extension exists and, therefore, no version negotiation is required with the client.

There is no negotiation of capabilities; however, the client advertises its capability to support the size of license data in the **wLicenseDetailLevel** field of the [Platform Challenge Response Data \(section 2.2.2.5.1\)](#) structure in the [Client Platform Challenge Response \(section 2.2.2.5\)](#).

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

The Remote Desktop Protocol: Licensing Extension packets are transported using TCP/IP.

2.2 Message Syntax

The following sections contain the Remote Desktop Protocol: Licensing Extension message syntax.

2.2.1 Common Data Structures

2.2.1.1 Security Headers

Each licensing message PDU contains one of the RDP security headers specified in [\[MS-RDPBCGR\]](#) section 2.2.8.1.1.2.

2.2.1.1.1 Basic (TS_SECURITY_HEADER)

For the Basic security header, see [\[MS-RDPBCGR\]](#) section 2.2.8.1.1.2.1.

2.2.1.1.2 Non-FIPS (TS_SECURITY_HEADER1)

For the non-FIPS security header, [\[MS-RDPBCGR\]](#) section 2.2.8.1.1.2.2.

2.2.1.1.3 FIPS (TS_SECURITY_HEADER2)

For the FIPS security header, see [\[MS-RDPBCGR\]](#) section 2.2.8.1.1.2.3.

2.2.1.2 Licensing Preamble (LICENSE_PREAMBLE)

For the licensing preamble, see [\[MS-RDPBCGR\]](#) section 2.2.1.12.1.1.

2.2.1.3 Licensing Binary BLOB (LICENSE_BINARY_BLOB)

The Licensing binary large object (BLOB) is specified in [\[MS-RDPBCGR\]](#) section 2.2.1.12.1.2.

2.2.1.4 Server Certificate (SERVER_CERTIFICATE)

The Server Certificate structure is specified in [\[MS-RDPBCGR\]](#) section 2.2.1.4.3.1. This structure holds either a server proprietary certificate (see [\[MS-RDPBCGR\]](#) section 2.2.1.4.3.1.1) or an X.509 certificate chain (see section [2.2.1.4.2](#)).

2.2.1.4.1 Server Proprietary Certificate (PROPRIETARYSERVERCERTIFICATE)

Proprietary certificates are specified in [\[MS-RDPBCGR\]](#) section 2.2.1.4.3.1.1.

2.2.1.4.2 X.509 Certificate Chain (X509_CERTIFICATE_CHAIN)

The X.509 Certificate Chain packet contains a collection of X.509 certificates.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
NumCertBlobs																															
CertBlobArray (variable)																															
...																															
Padding (variable)																															
...																															

NumCertBlobs (4 bytes): A 32-bit unsigned integer. This field specifies the number of [CertBlob](#) structures in the **CertBlobArray** field. The minimum value MUST be 2 (self-signed license server certificate and terminal server certificate) and the maximum value MUST be 200 (clearing house issued license server certificate chain and terminal server certificate).

CertBlobArray (variable): An array of CertBlob structures. If the **license server** was issued an X.509 certificate chain by the **clearing house**, this array contains all the certificates from that chain, in root-certificate-first order. The second-to-last element in the array is the **license server certificate**. The **terminal server certificate** is the last element in this array. If the license server certificate is self-signed, this array contains only two elements: the license server certificate and the terminal server certificate. The license server certificate is also the root certificate, if the license server certificate is self-signed.

Padding (variable): A byte array of the length $8 + 4 * \text{NumCertBlobs}$ is appended at the end the packet.

2.2.1.4.2.1 CertBlob (CERT_BLOB)

The CertBlob packet encapsulates an X.509 certificate (as specified in [RFC3280](#) section 4).

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
cbCert																															
abCert (variable)																															
...																															

cbCert (4 bytes): A 32-bit unsigned integer. This field specifies the number of bytes in **abCert**.

abCert (variable): A byte array of length **cbCert**. This field contains binary data representing a single X.509 certificate.

2.2.1.4.3 Proprietary Certificate (PROPRIETARYSERVERCERTIFICATE)

For proprietary certificates, see [MS-RDPBCGR](#) section 2.2.1.4.3.1.1.

2.2.2 Licensing PDU (TS_LICENSING_PDU)

The Licensing PDU packet encapsulates licensing messages that are exchanged between a **client** and a **terminal server**.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
tpktHeader																															
x224Data																								mcsPDU (variable)							
...																															
securityHeader (variable)																															
...																															
preamble																															
LicensingMessage (variable)																															
...																															

tpktHeader (4 bytes): A TPKT header, as specified in [T123] section 8.

x224Data (3 bytes): An X.224 Class 0 Data TPDU, as specified in [X224] section 13.7.

mcsPDU (variable): If the PDU is being sent from the client to the server, this field MUST contain a variable-length PER-encoded MCS Send Data Request PDU, as specified in [T125] (the ASN.1 structure definition is specified in [T125] section 7 part 7). The **userData** field of the MCS Send Data Request PDU contains a security header, a [licensing preamble](#), and the licensing message.

If the PDU is being sent from the server to the client, this field MUST contain a variable-length PER-encoded MCS Send Data Indication PDU, as specified in [T125] (the ASN.1 structure definition is specified in [T125] section 7 part 7). The **userData** field of the MCS Send Data Indication PDU contains a security header, a licensing preamble, and the licensing message.

securityHeader (variable): A security header. This field contains one of the following headers.

If the PDU is being sent from the client to the server:

- The **securityHeader** field SHOULD contain at least a [Basic](#) security header (see [MS-RDPBCGR] section 2.2.8.1.1.2.1). If the embedded flags field contains the SEC_ENCRYPT(0x0008) flag, and the Encryption Level selected by the server (see sections 5.1.2 and 2.2.1.4.3) is ENCRYPTION_LEVEL_LOW (1), ENCRYPTION_LEVEL_CLIENT_COMPATIBLE (2), or ENCRYPTION_LEVEL_HIGH (3), the securityHeader SHOULD contain a [non-FIPS](#) security header.

If the PDU is being sent from the server to the client, then the format of the security header depends on the Encryption Level and Encryption Method selected by the server (see [MS-RDPBCGR] section 5.3.2 and 2.2.1.4.3). This field MUST contain one of the following headers:

- Basic Security Header (see [MS-RDPBCGR] section 2.2.8.1.1.2.1) if the Encryption Level selected by the server (see [MS-RDPBCGR] section 5.3.2 and 2.2.1.4.3) is ENCRYPTION_LEVEL_NONE (0) or ENCRYPTION_LEVEL_LOW (1) and the embedded flags field does not contain the SEC_ENCRYPT (0x0008) flag.
- A non-FIPS Security Header (see [MS-RDPBCGR] section 2.2.8.1.1.2.2) if the Encryption Level selected by the server (see [MS-RDPBCGR] section 5.3.2 and 2.2.1.4.3) is

ENCRYPTION_LEVEL_CLIENT_COMPATIBLE (2) or ENCRYPTION_LEVEL_HIGH (3) and the embedded flags field contains the SEC_ENCRYPT (0x0008) flag.

- A [FIPS](#) Security Header (see [MS-RDPBCGR] section 2.2.8.1.1.2.3) if the Encryption Level selected by the server (see [MS-RDPBCGR] section 5.3.2 and 2.2.1.4.3) is ENCRYPTION_LEVEL_FIPS (4) and the embedded flags field contains the SEC_ENCRYPT (0x0008) flag.

If the Encryption Level is set to ENCRYPTION_LEVEL_CLIENT_COMPATIBLE (2), ENCRYPTION_LEVEL_HIGH (3), or ENCRYPTION_LEVEL_FIPS (4) and the flags field of the Security Header does not contain the SEC_ENCRYPT (0x0008) flag (the licensing PDU is not encrypted), then the field **MUST** contain a Basic Security Header (see [MS-RDPBCGR] section 2.2.8.1.1.2.1).

The SEC_LICENSE_ENCRYPT_CS (0x0200) and SEC_LICENSE_ENCRYPT_SC (0x0200) flags are used to communicate whether encryption is to be applied to the licensing PDUs (see [MS-RDPBCGR] section 2.2.8.1.1.2.1).

The **flags** field of the security header **MUST** contain the SEC_LICENSE_PKT (0x0080) flag (see [MS-RDPBCGR] section 2.2.8.1.1.2.1) for all the licensing messages.

preamble (4 bytes): A licensing preamble (see [MS-RDPBCGR] section 2.2.1.12.1.1) structure containing header information. The **bMsgType** field of the **preamble** structure specifies the type of the licensing message that follows the **preamble**.

The **bVersion** field of the **preamble** structure specifies the license protocol version and the client capability to handle extended error information in the Low nibble and High nibble respectively ([MS-RDPBCGR] section 2.2.1.12.1.1).

LicensingMessage (variable): A variable-length licensing message whose structure depends on the value of the **bMsgType** field in the **preamble** structure. The following table lists possible values for **bMsgType** and the associated licensing message (this table also appears in [MS-RDPBCGR] section 2.2.1.12.1.1).

Sent by the server.

Value	Meaning
LICENSE_REQUEST 0x01	The Licensing PDU is a License Request PDU, and the LicensingMessage contains a Server License Request .
PLATFORM_CHALLENGE 0x02	The Licensing PDU is a Platform Challenge PDU, and the LicensingMessage contains a Server Platform Challenge .
NEW_LICENSE 0x03	The Licensing PDU is a New License PDU, and the LicensingMessage contains a Server New License structure.
UPGRADE_LICENSE 0x04	The Licensing PDU is an Upgrade License PDU, and the LicensingMessage contains a Server Upgrade License structure.

Sent by the client.

Value	Meaning
LICENSE_INFO 0x12	The Licensing PDU is a License Info PDU, and the LicensingMessage contains a Client License Information structure.
NEW_LICENSE_REQUEST 0x13	The Licensing PDU is a New License Request PDU, and the LicensingMessage contains a Client New License Request structure.

Value	Meaning
PLATFORM_CHALLENGE_RESPONSE 0x15	The Licensing PDU is a Platform Challenge Response PDU, and the LicensingMessage contains a Client Platform Challenge Response structure.

Sent by either the client or the server.

Value	Meaning
ERROR_ALERT 0xFF	The Licensing PDU is a Licensing Error Message PDU, and the LicensingMessage contains a license error message structure.

2.2.2.1 Server License Request (SERVER_LICENSE_REQUEST)

The Server License Request packet is sent to the client to initiate the RDP licensing handshake.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ServerRandom (32 bytes)																															
...																															
...																															
ProductInfo (variable)																															
...																															
KeyExchangeList (variable)																															
...																															
ServerCertificate (variable)																															
...																															
ScopeList (variable)																															
...																															

ServerRandom (32 bytes): A 32-byte array containing a random number. This random number is created using a cryptographically secure pseudo-random number generator and is used to generate licensing encryption keys (see section [5.1.3](#)). These keys are used to encrypt licensing data in subsequent licensing messages (see sections [5.1.4](#) and [5.1.5](#)).

ProductInfo (variable): A variable-length [Product Information](#) structure. This structure contains the details of the product license required for connecting to the **terminal server**.

KeyExchangeList (variable): A [Licensing Binary BLOB](#) structure (see [\[MS-RDPBCGR\]](#) section 2.2.1.12.1.2) of type BB_KEY_EXCHG_ALG_BLOB (0x000D). This BLOB contains the list of 32-bit

unsigned integers specifying key exchange algorithms that the server supports. The terminal server supports only one key exchange algorithm as of now, so the BLOB contains the following value.

Value	Meaning
KEY_EXCHANGE_ALG_RSA 0x00000001	Indicates RSA key exchange algorithm with a 512-bit asymmetric key. <2>

ServerCertificate (variable): A Licensing Binary BLOB structure (see [MS-RDPBCGR] section 2.2.1.12.1.2) of type BB_CERTIFICATE_BLOB (0x0003). This BLOB contains the terminal server certificate (see section [2.2.1.4](#)). The terminal server can choose not to send the certificate by setting the **wbBlobLen** field in the Licensing Binary BLOB structure to 0. If encryption is in effect and is already protecting RDP traffic, the licensing protocol MAY [<3>](#) choose not to send the server certificate (for RDP security measures, see [MS-RDPBCGR] sections 5.3 and 5.4). If the licensing protocol chooses not to send the server certificate, then the client uses the public key obtained from the server certificate sent as part of Server Security Data in the Server MCS Connect Response PDU (see [MS-RDPBCGR] section 2.2.1.4).

ScopeList (variable): A variable-length [Scope List](#) structure that contains a list of entities that issued the **client license**. This list is used by the client in conjunction with **ProductInfo** to search for an appropriate license in its **license store**. [<4>](#)

2.2.2.1.1 Product Information (PRODUCT_INFO)

The Product Information packet contains the details of the product license that is required for connecting to the **terminal server**. The client uses this structure together with the [scope list](#) to search for and identify an appropriate license in its **license store**. Depending on the outcome of the search, the client sends a [Client New License Request](#) (section 2.2.2.2), [Client License Information](#) packet (section 2.2.2.3), or [license error message](#) (section 2.2.2.7.1) to the server.

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
dwVersion																															
cbCompanyName																															
pbCompanyName (variable)																															
...																															
cbProductId																															
pbProductId (variable)																															
...																															

dwVersion (4 bytes): A 32-bit unsigned integer that contains the license version information. The high-order word contains the major version of the operating system on which the terminal server is running, while the low-order word contains the minor version. [<5>](#)

cbCompanyName (4 bytes): An unsigned 32-bit integer that contains the number of bytes in the **pbCompanyName** field, including the terminating null character. This value MUST be greater than zero.

pbCompanyName (variable): Contains a null-terminated **Unicode string** that specifies the company name. [<6>](#)

cbProductId (4 bytes): An unsigned 32-bit integer that contains the number of bytes in the **pbProductId** field, including the terminating null character. This value **MUST** be greater than zero.

pbProductId (variable): Contains a null-terminated Unicode string that identifies the type of the license that is required by the terminal server. It **MAY** have the following string value.

Value	Meaning
"A02"	Per device or per user license

2.2.2.1.2 Scope List (SCOPE_LIST)

The Scope List packet contains a list of entities that issued a **client license**. The client uses the name of the issuers in the [Scope](#) structures of this list in conjunction with the [Product Information](#) structure to search the **license store** for a matching client license.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ScopeCount																															
ScopeArray (variable)																															
...																															

ScopeCount (4 bytes): A 32-bit unsigned integer containing the number of elements in the **ScopeArray** field.

ScopeArray (variable): An array of Scope structures containing **ScopeCount** elements. [<7>](#)

2.2.2.1.2.1 Scope (SCOPE)

The Scope packet contains the name of an entity that issued a **client license**.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Scope (variable)																															
...																															

Scope (variable): A Licensing Binary BLOB structure (see [\[MS-RDPBCGR\]](#) section 2.2.1.12.1.2) of type BB_SCOPE_BLOB (0x000E). This BLOB contains the name of a license issuer in null-terminated ANSI characters, as specified in [\[ISO/IEC-8859-1\]](#), string format, with an implementation-specific valid code page.

2.2.2.2 Client New License Request (CLIENT_NEW_LICENSE_REQUEST)

The Client New License Request packet is sent to a server when the client cannot find a license matching the product information provided in the [Server License Request](#) message. This message is

interpreted as a new license request by the server, and the server SHOULD attempt to issue a new license to the client on receipt of this message.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
PreferredKeyExchangeAlg																															
PlatformId																															
ClientRandom (32 bytes)																															
...																															
...																															
EncryptedPreMasterSecret (variable)																															
...																															
ClientUserName (variable)																															
...																															
ClientMachineName (variable)																															
...																															

PreferredKeyExchangeAlg (4 bytes): A 32-bit unsigned integer that indicates the key exchange algorithm chosen by the client. It MUST be set to KEY_EXCHANGE_ALG_RSA (0x00000001), which indicates an RSA-based key exchange with a 512-bit asymmetric key. [<8>](#)

PlatformId (4 bytes): A 32-bit unsigned integer. This field is composed of two identifiers: the operating system identifier and the independent software vendor (ISV) identifier. The platform ID is composed of the logical OR of these two values.

The most significant byte of the **PlatformId** field contains the operating system version of the client. [<9>](#)

The second most significant byte of the **PlatformId** field identifies the ISV that provided the client image. [<10>](#)

The remaining two bytes in the **PlatformId** field are used by the ISV to identify the build number of the operating system. [<11>](#)

ClientRandom (32 bytes): A 32-byte random number generated by the client using a cryptographically secure pseudo-random number generator. The **ClientRandom** and **ServerRandom** (see section 2.2.2.1) values, along with the data in the **EncryptedPreMasterSecret** field, are used to generate licensing encryption keys (see section [5.1.3](#)). These keys are used to encrypt licensing protocol messages (see sections [5.1.4](#) and [5.1.5](#)).

EncryptedPreMasterSecret (variable): A Licensing Binary BLOB structure (see [\[MS-RDPBCGR\]](#) section 2.2.1.12.1.2) of type BB_RANDOM_BLOB (0x0002). This BLOB contains an encrypted 48-byte random number. For instructions on how to encrypt this random number, see section [5.1.2.1](#).

ClientUserName (variable): A Licensing Binary BLOB structure (see [MS-RDPBCGR] section 2.2.1.12.1.2) of type BB_CLIENT_USER_NAME_BLOB (0x000F). This BLOB contains the client user name string in null-terminated **ANSI character set** format and is used along with the **ClientMachineName** BLOB to keep track of licenses issued to clients.

ClientMachineName (variable): A Licensing Binary BLOB structure (see [MS-RDPBCGR] section 2.2.1.12.1.2) of type BB_CLIENT_MACHINE_NAME_BLOB (0x0010). This BLOB contains the client machine name string in null-terminated ANSI character set format and is used along with the **ClientUserName** BLOB to keep track of licenses issued to clients.

2.2.2.3 Client License Information (CLIENT_LICENSE_INFO)

The Client License Information packet is sent by a client that already has a license issued to it in response to the [Server License Request \(section 2.2.2.1\)](#) message.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
PreferredKeyExchangeAlg																															
PlatformId																															
ClientRandom (32 bytes)																															
...																															
...																															
EncryptedPreMasterSecret (variable)																															
...																															
LicenseInfo (variable)																															
...																															
EncryptedHWID (variable)																															
...																															
MACData (16 bytes)																															
...																															
...																															

PreferredKeyExchangeAlg (4 bytes): The content and format of this field are the same as the **PreferredKeyExchangeAlg** field of the [Client New License Request \(section 2.2.2.2\)](#) message.

PlatformId (4 bytes): The content and format of this field are the same as the **PlatformId** field of the Client New License Request message.

ClientRandom (32 bytes): The content and format of this field are the same as the **ClientRandom** field of the Client New License Request message.

EncryptedPreMasterSecret (variable): The content and format of this field are the same as the **EncryptedPreMasterSecret** field of the Client New License Request message.

LicenseInfo (variable): A Licensing Binary BLOB structure (see [\[MS-RDPBCGR\]](#) section 2.2.1.12.1.2) of type BB_DATA_BLOB (0x0001). This BLOB contains the CAL (see the **pbLicenseInfo** field in section [2.2.2.6.1](#)) that is retrieved from the client's **license store**.

EncryptedHWID (variable): A Licensing Binary BLOB structure (see [\[MS-RDPBCGR\]](#) section 2.2.1.12.1.2). This BLOB contains a [Client Hardware Identification \(section 2.2.2.3.1\)](#) structure encrypted with the licensing encryption keys (see section [5.1.3](#)), using **RC4** (for instructions on how to perform the encryption, see section [5.1.4](#)).

MACData (16 bytes): An array of 16 bytes containing an **MD5 digest (Message Authentication Code (MAC))** that is generated over the unencrypted Client Hardware Identification structure. For instructions on how to generate this message digest, see section [5.1.6](#); for a description of how the server uses the **MACData** field to verify the integrity of the Client Hardware Identification structure, see section [3.1.5.1](#).

2.2.2.3.1 Client Hardware Identification (CLIENT_HARDWARE_ID)

The Client Hardware Identification packet is used for uniquely identifying a **Remote Desktop client** for the purpose of issuing a license. A **license server** uses the content of this structure as an index into the issued licenses in its database.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
PlatformId																															
Data1																															
Data2																															
Data3																															
Data4																															

PlatformId (4 bytes): The content and format of this field are the same as the **PlatformId** field of the [Client New License Request](#).

Data1 (4 bytes): A 32-bit unsigned integer containing client hardware-specific data. This field MUST contain a number that helps the server uniquely identify the client.

Data2 (4 bytes): A 32-bit unsigned integer containing client hardware-specific data. This field MUST contain a number that helps the server uniquely identify the client.

Data3 (4 bytes): A 32-bit unsigned integer containing client hardware-specific data. This field MUST contain a number that helps the server uniquely identify the client.

Data4 (4 bytes): A 32-bit unsigned integer containing client hardware-specific data. This field MUST contain a number that helps the server uniquely identify the client.

2.2.2.4 Server Platform Challenge (SERVER_PLATFORM_CHALLENGE)

The Server Platform Challenge packet is sent from the server to the client after receiving the [Client New License Request \(section 2.2.2.2\)](#) or certain cases of [Client License Information \(section 2.2.2.3\)](#). For more information on Client License Information and when Server Platform Challenge is sent, see [Processing Client License Information \(section 3.2.5.3\)](#).

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ConnectFlags																															
EncryptedPlatformChallenge (variable)																															
...																															
MACData (16 bytes)																															
...																															
...																															

ConnectFlags (4 bytes): Reserved.

EncryptedPlatformChallenge (variable): A Licensing Binary BLOB<12> structure (see [\[MS-RDPBCGR\]](#) section 2.2.1.12.1.2). This BLOB contains the encrypted server platform challenge data. The server platform challenge data is a random string generated by the server and is encrypted with the licensing encryption key (see section [5.1.3](#)) using **RC4** (for instructions on how to perform the encryption, see section [5.1.4](#)).

MACData (16 bytes): An array of 16 bytes containing an **MD5 digest (MAC)** generated over the unencrypted platform challenge BLOB. For instructions on how to generate this message digest, see section [5.1.6](#); for a description of how the client uses the **MACData** field to verify the integrity of the platform challenge BLOB, see section [3.1.5.1](#).

2.2.2.5 Client Platform Challenge Response (CLIENT_PLATFORM_CHALLENGE_RESPONSE)

The Client Platform Challenge Response packet is sent by the client in response to the [Server Platform Challenge \(section 2.2.2.4\)](#) message.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
EncryptedPlatformChallengeResponse (variable)																															
...																															
EncryptedHWID (variable)																															
...																															
MACData (16 bytes)																															

...
...

EncryptedPlatformChallengeResponse (variable): A LICENSE_BINARY_BLOB<13> structure (as specified in [MS-RDPBCGR] section 2.2.1.12.1.2) of **wBlobType** BB_ENCRYPTED_DATA_BLOB (0x0009). This BLOB contains the encrypted [Platform Challenge Response Data \(section 2.2.2.5.1\)](#) generated by the client and is encrypted with the licensing encryption key (see section 5.1.3), using **RC4** (for instructions on how to perform the encryption, see section 5.1.4).

EncryptedHWID (variable): A LICENSE_BINARY_BLOB structure (as specified in [MS-RDPBCGR] section 2.2.1.12.1.2) of **wBlobType** BB_ENCRYPTED_DATA_BLOB (0x0009). This BLOB contains the encrypted [Client Hardware Identification \(section 2.2.2.3.1\)](#) and is encrypted with the licensing encryption key (see section 5.1.3) using RC4 (for instructions on how to perform the encryption, see section 5.1.4).

MACData (16 bytes): An array of 16 bytes containing an **MD5 digest (MAC)** generated over the decrypted Client Hardware Identification and Platform Challenge Response Data. For instructions on how to generate this message digest, see section 5.1.6; for a description of how the server uses the **MACData** field to verify the integrity of the Client Hardware Identification and the Platform Challenge Response Data, see section 3.1.5.1.

2.2.2.5.1 Platform Challenge Response Data (PLATFORM_CHALLENGE_RESPONSE_DATA)

The Platform Challenge Response Data packet contains information pertaining to the client's license handling capabilities and the Client Platform Challenge data sent by the server in the [Server Platform Challenge](#).

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
wVersion																wClientType															
wLicenseDetailLevel																cbChallenge															
pbChallenge (variable)																															
...																															

wVersion (2 bytes): A 16-bit unsigned integer that contains the platform challenge version. This field **MUST** be set to 0x0100.

wClientType (2 bytes): A 16-bit unsigned integer that represents the operating system type of the client and **MAY** contain one of following values.<14>

Value	Meaning
WIN32_PLATFORMCHALLENGE_TYPE 0x0100	Win32 Platform Challenge Type.
WIN16_PLATFORMCHALLENGE_TYPE 0x0200	Win16 Platform Challenge Type.
WINCE_PLATFORMCHALLENGE_TYPE	WinCE Platform Challenge Type.

Value	Meaning
0x0300	
OTHER_PLATFORMCHALLENGE_TYPE 0xFF00	Other Platform Challenge Type.

wLicenseDetailLevel (2 bytes): A 16-bit unsigned integer. This field represents the capability of the client to handle license data. RDP version 5.0 and later clients SHOULD advertise support for large (6.5 KB or higher) licenses by setting the detail level to LICENSE_DETAIL_DETAIL (0x0003). The following table lists valid values for this field.

Value	Meaning
LICENSE_DETAIL_SIMPLE 0x0001	License Detail Simple (client license certificate and license server certificate without issuer details).
LICENSE_DETAIL_MODERATE 0x0002	License Detail Moderate (client license certificate chain up to license server's certificate issuer).
LICENSE_DETAIL_DETAIL 0x0003	License Detail Detail (client license certificate chain up to root certificate).

cbChallenge (2 bytes): A 16-bit unsigned integer that indicates the number of bytes of binary data contained in the **pbChallenge** field.

pbChallenge (variable): Contains the decrypted Client Platform Challenge data sent by the server in the Server Platform Challenge message.

2.2.2.6 Server Upgrade License (SERVER_UPGRADE_LICENSE)

The Server Upgrade License packet is sent from the server to the client if the client presents an existing license and the server determines that this license SHOULD be upgraded. This message contains the upgraded license information, which the client uses to replace the previously held license in the client's **license store**.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
EncryptedLicenseInfo (variable)																															
...																															
MACData (16 bytes)																															
...																															
...																															

EncryptedLicenseInfo (variable): A LICENSE_BINARY_BLOB structure (as specified in [\[MS-RDPBCGR\]](#) section 2.2.1.12.1.2) of **wBlobType** BB_ENCRYPTED_DATA_BLOB (0x0009). **<15>** This BLOB contains the encrypted [New License Information \(section 2.2.2.6.1\)](#) packet and is encrypted with the licensing encryption key (see section [5.1.3](#)), using **RC4** (for instructions on how to perform the encryption, see section [5.1.4](#)).

MACData (16 bytes): An array of 16 bytes containing an **MD5 digest (Message Authentication Code)** generated over the unencrypted New License Information structure. For instructions on how to generate this message digest, see section [5.1.6](#); for a description of how the server uses the **MACData** field to verify the integrity of the New License Information packet, see section [3.1.5.1](#).

2.2.2.6.1 New License Information (NEW_LICENSE_INFO)

The New License Information packet contains the actual **client license** and associated indexing information. The client stores the license in its **license store** using the indexing information, and uses it in subsequent connections. The **dwVersion**, **pbScope**, **pbCompanyName**, and **pbProductId** fields are used by the client to index the licenses in the client's license store.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
dwVersion																															
cbScope																															
pbScope (variable)																															
...																															
cbCompanyName																															
pbCompanyName (variable)																															
...																															
cbProductId																															
pbProductId (variable)																															
...																															
cbLicenseInfo																															
pbLicenseInfo (variable)																															
...																															

dwVersion (4 bytes): The content and format of this field are the same as the **dwVersion** field of the [Product Information \(section 2.2.2.1.1\)](#) structure.

cbScope (4 bytes): A 32-bit unsigned integer that contains the number of bytes in the string contained in the **pbScope** field.

pbScope (variable): Contains the NULL-terminated **ANSI character set** string giving the name of the issuer of this license. For example, for licenses issued by TailSpin Toys, this field contains the string "TailSpin Toys".

cbCompanyName (4 bytes): The content and format of this field are the same as the **cbCompanyName** field of the Product Information structure.

pbCompanyName (variable): The content and format of this field are the same as the **pbCompanyName** field of the Product Information structure.

cbProductId (4 bytes): The content and format of this field are the same as the **cbProductId** field of the Product Information structure.

pbProductId (variable): The content and format of this field are the same as the **pbProductId** field of the Product Information structure.

cbLicenseInfo (4 bytes): A 32-bit unsigned integer that contains the number of bytes of binary data in the **pbLicenseInfo** field.

pbLicenseInfo (variable): This field contains the CAL issued to the client by the license server. This license consists of an X.509 certificate chain generated by the license server. The binary data contained in this field is opaque to the client. The client sends this information back to the server in the [Client License Information](#) message.

2.2.2.7 Server New License (SERVER_NEW_LICENSE)

The Server New License message is sent from the server to the client when a new license is issued to the client. The structure and the content of this message are the same as the [Server Upgrade License](#) message.

2.2.2.7.1 License Error Message (LICENSE_ERROR_MESSAGE)

The license error message specified in [\[MS-RDPBCGR\]](#) section 2.2.1.12.1.3 can be used by both client and server.

If the client supports extended error, the **terminal server** includes information relevant to the error code in the **bbErrorInfo** field of the Licensing Error Message. For more details, see [\[MS-RDPBCGR\]](#) section 2.2.1.12.1.3.

3 Protocol Details

3.1 Common Details

3.1.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

3.1.2 Timers

None.

3.1.3 Initialization

None.

3.1.4 Higher-Layer Triggered Events

None.

3.1.5 Message Processing Events and Sequencing Rules

3.1.5.1 Message Integrity Checking

Both the **server** and **client** add a **MAC** checksum to licensing messages to allow the recipient to validate the integrity of the licensing data that is contained in the message.

The sender **MUST** generate the MAC checksum (see section [5.1.6](#)) on selected fields of a licensing message (for fields that are used to generate the MAC checksum, see the **MACData** fields in sections [2.2.2.3](#) through [2.2.2.6](#)). It then **MUST** encrypt those fields of the licensing message (see [Encrypting Licensing Session Data \(section 5.1.4\)](#)). Next, it **MUST** transmit the licensing message (see [Licensing PDU \(section 2.2.2\)](#)) containing the encrypted fields and the MAC checksum to the receiver.

The receiver **MUST** decrypt the encrypted fields of the received licensing message (see [Decrypting Licensing Session Data \(section 5.1.5\)](#)), and then **MUST** generate a MAC checksum over the decrypted fields. Next, it **MUST** compare the generated checksum with the received checksum. If they do not match, The receiver **MAY** send a [Licensing Error Message](#) with an error code `ERR_INVALID_MAC` and a state transition code of `ST_TOTAL_ABORT` to the sender.

3.1.5.2 Sending License Error Messages

Both the client and the server can send a [license error message](#). Whenever an error message is sent, the message type in the [licensing preamble](#) **MUST** be set to `ERROR_ALERT` (0xFF). For the PDU, see [\[MS-RDPBCGR\]](#) section 2.2.1.12.1.3

The client and the server **MUST** also set the appropriate state transition value in the **dwStateTransition** field in the PDU. This is used to determine the next action to take. For state transitions, see [Processing License Error Messages](#).

A more detailed reason for the error MAY be passed by using the **bbErrorInfo** BLOB. The BLOB type MUST be BB_ERROR_BLOB (see [MS-RDPBCGR] sections 2.2.1.12.1.2 and 2.2.1.12.1.3). This BLOB is empty if no detailed reason for the error is passed.

3.1.5.3 Processing License Error Messages

Both the server and the client can send a [license error message](#) and indicate a state transition with the error code. Possible state transitions include the following:

- ST_TOTAL_ABORT aborts the licensing protocol. The server and the client can both send this state transition. Aborting the licensing protocol MAY disconnect the RDP client connection, depending on the server licensing system. [16](#)
- ST_NO_TRANSITION is used when the server is to indicate success. It MUST set the **dwErrCode** field to STATUS_VALID_CLIENT and state transition to ST_NO_TRANSITION (see [MS-RDPBCGR](#) section 2.2.1.12.1.3).
- ST_RESET_PHASE_TO_START resets the licensing protocol and client goes back to "CLIENT LICENSING AWAIT" state and server goes back to "SERVER LICENSING BEGIN" state.
- ST_RESEND_LAST_MESSAGE makes the client and server resend the previously sent message.

ST_RESET_PHASE_TO_START and ST_RESEND_LAST_MESSAGE are not shown in the Client State Transition diagram (section [3.1.5.3.1](#)) and Server State Transition diagram (section [3.1.5.3.2](#)), as they can cause confusion. [17](#)

3.1.5.3.1 Client State Transition

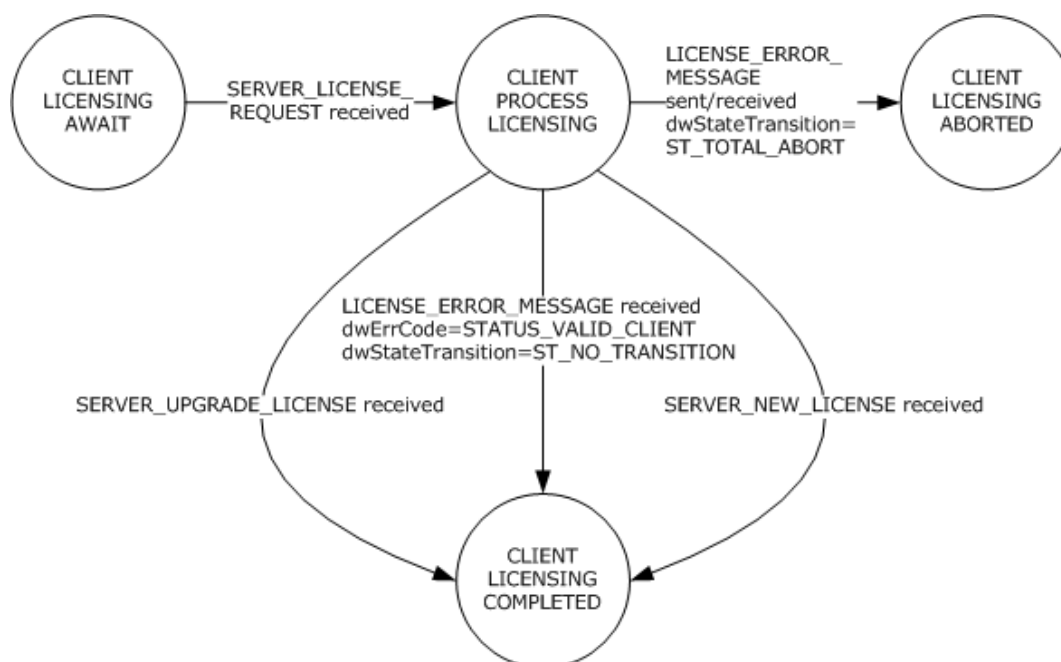


Figure 5: Client state transition

3.1.5.3.2 Server State Transition

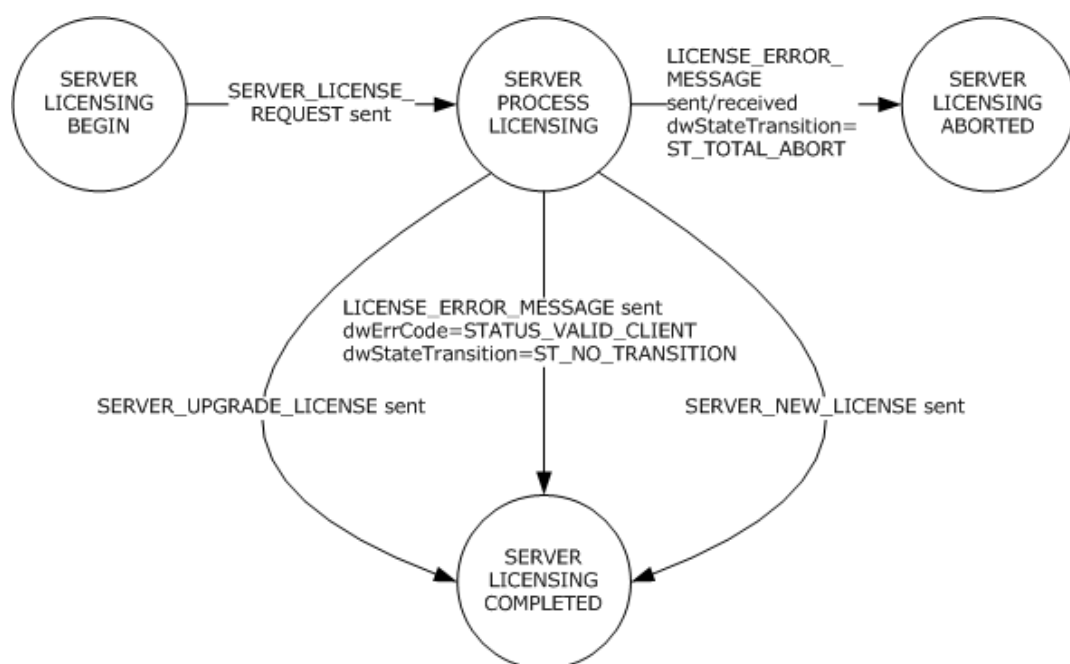


Figure 6: Server state transition

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

3.2 Server Details

3.2.1 Abstract Data Model

Refer to the common details abstract data model in section [3.1.1](#).

3.2.1.1 Server Random

The Server Random is a 32-byte random number created using a cryptographically secure pseudo-random number generator. It is used to generate encryption keys that are used in later stages of the licensing protocol; see [Server License Request \(section 2.2.2.1\)](#).

The server random is unique for a given client connection. It is created at the beginning of the licensing protocol and is destroyed after the licensing protocol is completed. For information on how the Server Random is used, see sections 2.2.2.1 and [5.1.2](#).

3.2.1.2 Product Information

The [Product Information](#) packet contains the operating system version, company name, and product ID. This information is used by the client to search for a previously issued CAL stored in the client's **license store**.

For example, the content of the Product Information structure can be as follows.

```
dwVersion = 0x00050002
pbCompanyName = "Microsoft Corporation"
pbProductId = "A02"
```

3.2.1.3 Server Certificate

A [server certificate](#) MUST be either a [proprietary certificate](#) or an X.509 certificate. An X.509 certificate chain is issued to a **terminal server** by a **license server**. If a terminal server has not received an X.509 certificate from a license server, it generates a proprietary certificate and sends it in the **ServerCertificate** field of a [Server License Request](#) message.

3.2.1.4 Key Exchange List

The Key Exchange List is a list of key exchange algorithms supported by the server. See the **KeyExchangeList** field in the [Server License Request](#) message.

3.2.1.5 Scope List

The [Scope List](#) describes the list of issuers for the CAL.<18>

3.2.1.6 Platform Challenge

The platform challenge is a random string generated by the server. This string is encrypted (see [Encrypting Licensing Data \(section 5.1.4\)](#)) with the licensing encryption key using **RC4** and sent in the **EncryptedPlatformChallenge** field of the [Server Platform Challenge](#) message. It is created at the beginning of the licensing protocol and destroyed when the licensing protocol is completed.

3.2.1.7 License

The license is an X.509 certificate chain that contains a certificate issued by the **license server** to the client. The leaf X.509 certificate in the certificate chain has the following information:

- Version: The version of the **terminal server** that this client is allowed to access.
- Type: Whether the license is temporary or permanent.
- Validity Period: The period of validity of this license.

The server MUST send the license to the client in the [Server New License](#) message or the [Server Upgrade License](#) message.

3.2.1.8 ClientUserName

The **ClientUserName** is the name of the user initiating the remote connection to the **terminal server**. The **ClientUserName** is sent from the client to the terminal server in the CLIENT_NEW_LICENSE_REQUEST message (section [2.2.2.2](#)).

3.2.1.9 ClientMachineName

The **ClientMachineName** is the name of the device from which the remote connection is made to the terminal server. This information is sent from the client to the terminal server in the CLIENT_NEW_LICENSE_REQUEST message (section [2.2.2.2](#)).

3.2.1.10 Encryption Keys

The server uses the 128-bit licensing encryption key to encrypt and to decrypt licensing information message data obtained in Server Platform Challenge messages (section [2.2.2.4](#)), in Client HWID (section [2.2.2.3](#) and [2.2.2.3.1](#)), and in Client Platform Challenge messages (section [2.2.2.5](#)), as specified in section [5.1.3](#).

3.2.1.11 Server Licensing States

Server Licensing States is an enumeration of different licensing states that the server can have. Server licensing state transition is shown in the diagram in section [3.1.5.3.2](#). The following are the possible licensing states:

Server Licensing Begin: The **terminal server** licensing protocol starts in the "Server Licensing Begin" state.

Server Process Licensing: A successful [SERVER_LICENSE_REQUEST](#) (section [2.2.2.1](#)) call brings the server to the "Server Process Licensing" state. In this state, the terminal server processes the [Client New License Requests](#) or [Client License Information](#) (as specified in sections [3.2.5.2](#) and [3.2.5.3](#)) depending on the availability of the requested license on the client.

Server Licensing Aborted: When the client sends Out-of-Sequence or Unrecognized Messages or Invalid MAC (as specified in sections [3.2.5.8](#) and [3.2.5.9](#)), the terminal server goes into "Server Licensing Aborted" state.

Server Licensing Completed: When a new license is sent to the remote client (as specified in section [3.2.5.7](#)), or the **client license** is upgraded and sent to the remote client (as specified in section [3.2.5.6](#)), or the remote client doesn't need any license to connect (as specified in section [3.2.5.2](#)), the terminal server goes into "Server Licensing Completed" state.

3.2.2 Timers

None. The licensing protocol does not have its own timers. Events such as connection timeout are handled by RDPBCGR (see [\[MS-RDPBCGR\]](#) section 3.3.6.1).

3.2.3 Initialization

The server licensing state MUST be initialized to "Server Licensing Begin".

3.2.4 Higher-Layer Triggered Events

None.

3.2.5 Message Processing Events and Sequencing Rules

3.2.5.1 Sending Server License Request PDUs

The server initiates the licensing protocol by sending a [Server License Request](#) to the client. A Server License Request uses the message type of LICENSE_REQUEST (0x01) in the [licensing preamble](#).

3.2.5.2 Processing Client New License Requests

The client MUST send the [Client New License Request](#) (section [2.2.2.2](#)) when it does not have a license.

In case of a **personal terminal server**, no processing is done on the server side, and the server sends a license error message with the error code STATUS_VALID_CLIENT and the state transition code ST_NO_TRANSITION. The licensing protocol is complete at this point.

In the case of terminal servers, the server tries to follow the new license flow (section [1.3.3.1](#)).

The server MUST compute the **license encryption key** (see section [5.1.3](#)) by using the client-generated random number and **premaster secret**. The premaster secret is obtained by decrypting the encrypted premaster secret with the terminal server's private key. This allows elements of the remainder of the licensing protocol to be encrypted.

The terminal server MUST decrypt (see section [5.1.5](#)) the **EncryptedHWID** field (see section [2.2.2.3](#)) using the license encryption key to get the [Client Hardware Identification \(section 2.2.2.3.1\)](#) structure. It then MUST generate the **MAC** checksum (see section [5.1.6](#)) over the decrypted Client Hardware Identification and MUST compare it with the MAC checksum received in the Client License Information message to verify data integrity.

The server MUST respond by issuing a platform challenge to the client. The server MUST encrypt and MUST send the platform challenge in a [Server Platform Challenge \(section 2.2.2.4\)](#) message.

The **ClientUserName** and **ClientMachineName** fields are preserved for the licensing protocol session and are used to issue a CAL to the client when the client successfully responds to the Server Platform Challenge message.

3.2.5.3 Processing Client License Information

If the client has a valid license that matches the information sent by the server in the [Server License Request \(section 2.2.2.1\)](#) message, the client MUST send the previously issued license in the [Client License Information \(section 2.2.2.3\)](#) message.

In the case of a **personal terminal server**, the sent license is cached by the server, and then the server sends a license error message with error code STATUS_VALID_CLIENT and the state transition code ST_NO_TRANSITION. The licensing protocol is complete at this point.

In the case of terminal servers, the server tries to follow the upgrade license flow (section [1.3.3.2](#)).

The server MUST compute the **license encryption key** by using the client-generated random number and the **premaster secret**. The premaster secret is obtained by decrypting the encrypted premaster secret with the terminal server's private key. This allows elements of the remainder of the licensing protocol to be encrypted.

The terminal server MUST decrypt (see section [5.1.5](#)) the **EncryptedHWID** field (see section [2.2.2.3](#)) using the license encryption key to get the [Client Hardware Identification \(section 2.2.2.3.1\)](#) structure. It then MUST generate the **MAC** checksum (see section [5.1.6](#)) over the decrypted Client Hardware Identification and compares it with the MAC checksum received in the Client License Information message to verify data integrity.

The server MUST validate the [license](#) present in the LicenseInfo BLOB (see section [2.2.2.3](#)) by validating the following data:

- Valid **license server** signature.
- Correct product information.
- License expiration date.
- Client hardware identification.

The server MUST respond as follows, depending on the validity of the **client license**.

Case 1: The client presents a valid **permanent license** that does not require an upgrade.

The server MUST send a [license error](#) message with the error code STATUS_VALID_CLIENT and the state transition code ST_NO_TRANSITION. The licensing protocol is complete at this point.

Case 2: If the client presents a license that requires upgrading, that is:

- A valid permanent license that is expired.
- A valid permanent license that is about to expire (within seven days of its expiry).
- A valid **temporary license**.
- An invalid license.

The server MUST respond back with [Server Platform Challenge](#) message to the client.

3.2.5.4 Sending Server Platform Challenges

The [Server Platform Challenge](#) message MUST be sent from the **terminal server** when a new license needs to be issued to the client or when an existing **client license** needs to be upgraded. The message type is PLATFORM_CHALLENGE (0x02) in the [licensing preamble](#).

The server platform challenge is never sent when the target is a **personal terminal server**. This is because a personal terminal server never tries to get licenses issued or upgraded.

3.2.5.5 Processing Client Platform Challenge Responses

When a server receives the [Client Platform Challenge Response](#) message, it decrypts the **EncryptedPlatformChallengeResponse** and **EncryptedHWID** fields in the message using the **license encryption key** generated while processing earlier licensing messages.

The server MUST then generate the **MAC** checksum over the decrypted [Platform Challenge Response Data](#) packet and decrypted [Client Hardware Identification](#) packet, and MUST compare it with the received MAC checksum to verify the data integrity. [Handling invalid MACs \(section 3.2.5.9\)](#) is specified in section 3.2.5.9.

The following cases can result:

Case 1: A [Client License Information](#) message was received earlier by the server, and the CAL (LicenseInfo BLOB) in the message required an upgrade.

If the **license server** cannot be contacted to upgrade the license, and the old license is still valid, the terminal server sends the [Server Upgrade License](#) message and returns the old license to the client.

Case 2: If either of the following conditions occurs:

- The CAL (LicenseInfo BLOB) received in the Client License Information message required an upgrade, and the license server cannot be contacted to upgrade the CAL and the old license is not valid.

Or:

- A [Client New License Request](#) message was received earlier, and the license server cannot be contacted to issue a new CAL.

In this case, if the server's **grace period** has not been exceeded, the server responds as if the client presented a valid license by sending a [license error message](#) with an error code of STATUS_VALID_CLIENT (0x00000007) and a state transition code of ST_NO_TRANSITION (0x00000002), ending the licensing protocol.

If the server's grace period has been exceeded, it sends a license error message with error code `ERR_NO_LICENSE_SERVER` (0x00000006) and a state transition of `ST_TOTAL_ABORT` (0x00000001). The licensing protocol is aborted.

Case 3: If either of the following conditions occurs:

- The CAL (LicenseInfo BLOB) received in the Client License Information message required an upgrade, and the license server is available to upgrade the CAL, but it cannot upgrade the license and the old license is not valid.

Or:

- A Client New License Request message was received earlier, and the license server is available to issue a new CAL, but the server was not able to issue a new license.

In this case, if the grace period has not been exceeded, the server responds as if the client presented a valid license by sending a license error message with an error code of `STATUS_VALID_CLIENT` (0x00000007) and a state transition code of `ST_NO_TRANSITION` (0x00000002), ending the licensing protocol.

If the server's grace period has been exceeded, it sends a license error message with an error code of `ERR_INVALID_CLIENT` (0x00000008) and a state transition of `ST_TOTAL_ABORT` (0x00000001). The licensing protocol is aborted.

Case 4: The CAL (LicenseInfo BLOB) in the Client License Information message received by the server required an upgrade, and the CAL is valid and the license server available, but the license server cannot upgrade the license. In this case, the terminal server sends the Server Upgrade License message and returns the old license to the client.

Case 5: A Client License Information message was received earlier by the server; the CAL (LicenseInfo BLOB) in the message required an upgrade; the license server can be contacted; and the old license is successfully upgraded. In this case, the terminal server returns the upgraded CAL in a Server Upgrade License message.

Case 6: A Client New License Request message was received earlier, the license server was contacted, and it issued a new license. In this case, the terminal server sends the new license to the client in a [Server New License](#) message.

3.2.5.6 Sending Server Upgrade Licenses

The [Server Upgrade License](#) message MUST be sent to the client to upgrade a license in its **license store**. The message type is `UPGRADE_LICENSE` (0x04) in the [licensing preamble](#).

The **EncryptedLicenseInfo** field is the license information structure after encryption with the **license encryption key**, using **RC4**. The BLOB type is `BB_ENCRYPTED_DATA_BLOB` (0x0009). The **MACData** field is the 128-bit **MD5 digest** generated from the unencrypted upgraded license.

3.2.5.7 Sending Server New Licenses

The [Server New License](#) message MUST be sent to the client when a new license is required to be issued. The message type is `NEW_LICENSE` (0x03) in the [licensing preamble](#).

3.2.5.8 Handling Out-of-Sequence or Unrecognized Messages

If the server receives a message that is not expected according to the [Licensing PDU Flow](#), or a malformed or an unrecognized message, the server MUST send a [License Error message](#) with an error code of `ERR_INVALID_CLIENT` and a state transition code of `ST_TOTAL_ABORT`.

3.2.5.9 Handling Invalid MACs

If the **MAC** generated over decrypted fields of a message does not match the MAC contained in the message, the server MUST send a [License Error message](#) with an error code of ERR_INVALID_MAC and a state transition code of ST_TOTAL_ABORT.

3.2.6 Timer Events

None.

3.2.7 Other Local Events

None.

3.3 Client Details

3.3.1 Abstract Data Model

3.3.1.1 Platform ID

The **PlatformId** field is sent by the client in the [Client New License Request](#) message or the [Client License Information](#) message, depending on whether the client has already been issued a license. It is also sent by the client in the [Client Hardware Identification](#) structure. The **PlatformId** field content is populated by the client, as specified in Client New License Request (section 2.2.2.2). The **PlatformId** field is used by the server to uniquely identify a client in conjunction with the **Data1**, **Data2**, **Data3**, and **Data4** fields of the Client Hardware Identification structure.

3.3.1.2 Client Random

The **ClientRandom** field is a 32-byte random number created by using a cryptographically secure pseudo-random number generator (see sections [2.2.2.2](#) and [5.1.2](#)). It is created at the beginning of the licensing protocol and destroyed when the licensing protocol is completed.

3.3.1.3 Preferred Key Exchange Algorithm ID

The preferred key exchange algorithm ID is the key exchange algorithm selected by the client. This MUST be set to KEY_EXCHANGE_ALG_RSA (0x00000001), which indicates RSA with a 512-bit key (see [Client New License Request \(section 2.2.2.2\)](#)).<19>

3.3.1.4 Client User Name

See section [2.2.2.2](#).

3.3.1.5 Client Machine Name

See section [2.2.2.2](#).

3.3.1.6 Encrypted Premaster Secret

The encrypted premaster secret is a licensing BLOB containing a 48-byte random number that is encrypted using RSA with the server's public key retrieved from the [server certificate](#).

The **premaster secret** (along with client and server random values) is used to generate the **license encryption key** for later stages of the licensing protocol.

3.3.1.7 License

The license is a licensing BLOB of type BB_DATA_BLOB (0x0001). The server MUST send it to the client in the [Server New License](#) message or the [Server Upgrade License](#) message. It is stored by the client in the **license store**.

3.3.1.8 License Store

This is a client-side database used for storing **CALs**. This database is indexed by using the [product information](#) and the [scope list](#) associated with the license. The product information and the scope list are used for faster retrieval of the license by the client when processing the [Server License Request](#) message.

On receipt of the Server License Request message, the client searches its **license store** for a CAL. If the client has already been issued a CAL, it retrieves the binary data for the license from its license store by using the product information and the scope list as the lookup key. The client then MUST send it as a [Licensing Binary BLOB](#) in the [Client License Information](#) message.

3.3.1.9 Client Hardware Identification

The Client Hardware Identification is an identifier that uniquely identifies the client.

The content and format of the **PlatformID** field of [Client Hardware Identification](#) are the same as the **PlatformID** field of the [Client License Information](#) and [Client New License Request](#) messages. This ties a particular Client Hardware Identification to the client's operating system. The other 16 bytes (fields **Data1** through **Data4**) of the Client Hardware Identification are intended to be hardware-specific. Clients SHOULD attempt to use operating system-specific or hardware-specific information that is easily and consistently retrievable. Examples include hard-wired processor IDs, Ethernet addresses of nonremovable Ethernet cards, and disk subsystem serial numbers. The client SHOULD cache the Client Hardware Identification for later retrieval after it is generated.

3.3.1.10 Encryption Keys

The client uses the 128-bit licensing encryption key to encrypt and to decrypt licensing information message data obtained in Server Platform Challenge messages (section [2.2.2.4](#)), in Client HWID (section [2.2.2.3](#) and [2.2.2.3.1](#)), and in Client Platform Challenge messages (section [2.2.2.5](#)), as specified in section [5.1.3](#).

3.3.1.11 Client Licensing States

Client Licensing States is an enumeration of different licensing states that the remote client can have. Client licensing state transition is shown in the diagram in section [3.1.5.3.1](#).

Client Licensing Await: The client starts with the "Client Licensing Await" state.

Client Process Licensing: The client goes into the "Client Process Licensing" state after receiving [SERVER LICENSE REQUEST](#) (section [2.2.2.1](#)) (as specified in section [3.3.5.1](#)). In this state the remote client sends the license information (as specified in section [3.2.5.3](#)) or the new license request (as specified in section [3.2.5.2](#)), depending on the availability of the requested license on the client to the **terminal server**.

Client Licensing Aborted: When the terminal server sends Out-of-Sequence or Unrecognized Messages or Invalid MAC (as specified in sections [3.2.5.8](#) and [3.2.5.9](#)), the client goes into the "Server Licensing Aborted" state.

Client Licensing Completed: When a new license is received from the terminal server (as specified in section [3.3.5.7](#)) or on receiving the license that is upgraded from the terminal server (as specified

in section [3.3.5.6](#)) or on receiving error code STATUS_VALID_CLIENT and the state transition code ST_NO_TRANSITION from the terminal server, the client goes into the "Client Licensing Completed" state.

3.3.2 Timers

3.3.2.1 Client Packet Wait Timer

The client MAY [≤20>](#) implement a configurable Client Packet Wait Timer. If the client does not receive a response from the server for this time duration, the client MAY disconnect the RDP connection.

3.3.3 Initialization

The client licensing state MUST be initialized to "Client Licensing Await".

3.3.4 Higher-Layer Triggered Events

None.

3.3.5 Message Processing Events and Sequencing Rules

3.3.5.1 Processing Server License Requests

The server initiates the licensing protocol by sending a [Server License Request](#) message to the client.

The client MUST respond to the Server License Request in the following possible ways:

- If the [server certificate](#) does not authenticate correctly, the client MUST return a [license error message](#) with an error code of ERR_INVALID_SERVER_CERTIFICATE (0x01) and a state transition of ST_TOTAL_ABORT (0x01). The server MUST then end the licensing protocol.
- The client searches its **license store** to find a CAL that matches the [Product Information](#) packet provided in the Server License Request. If the client finds a matching license, it MUST respond with a [Client License Information](#) message.
- If the client does not find a license matching the product information provided in the Server License Request, it MUST request a new license by sending the [Client New License Request](#) message.
- The client MAY also choose to end the licensing protocol by sending a license error message with an error code of ERR_NO_LICENSE (0x02) and a state transition of ST_TOTAL_ABORT (0x01).

3.3.5.2 Sending Client New License Requests

The Sending Client New License Request message is sent when the client cannot find a license type in its **license store** that matches the [Product Information](#) given by the server in the [Server License Request](#) message.

The message type is NEW_LICENSE_REQUEST (0x13) in the [licensing preamble](#). See also [Client New License Request \(section 2.2.2.2\)](#).

3.3.5.3 Sending Client License Information

The [Client License Information](#) message MUST be sent when a client has a license in its **license store** that matches the [Product Information](#) sent in the [Server License Request](#) message.

The message type in the [licensing_preamble](#) is LICENSE_INFO (0x12).

3.3.5.4 Processing Server Platform Challenges

This message is sent from the server when a new license is required to be issued to the client or when the CAL presented by the client needs to be upgraded.

The client MUST decrypt the **EncryptedPlatformChallenge** field (see section [5.1.5](#)) received in the [Server Platform Challenge \(section 2.2.2.4\)](#) message using the **license encryption key** to obtain the platform challenge data. It then MUST generate a **MAC** checksum over the decrypted platform challenge data and compares it with the received MAC checksum to verify the integrity of the data.

The client MUST respond to the Server Platform Challenge with a [Client Platform Challenge Response \(section 2.2.2.5\)](#) message.

3.3.5.5 Sending Client Platform Challenge Responses

The client MUST send the [Client Platform Challenge Response \(section 2.2.2.5\)](#) message in response to a [Server Platform Challenge \(section 2.2.2.4\)](#) message. The message type in the [licensing_preamble](#) is PLATFORM_CHALLENGE_RESPONSE (0x15).

The client constructs a [Platform Challenge Response Data \(section 2.2.2.5.1\)](#) structure and encrypts it (see section [5.1.4](#)) by using the **license encryption key**. This encrypted Platform Challenge Response Data is sent as the EncryptedChallengeResponse BLOB of the Client Platform Challenge Response message.

3.3.5.6 Processing Server Upgrade Licenses

The client MUST receive a [Server Upgrade License \(section 2.2.2.6\)](#) message when the CAL sent by the client in the [Client License Information \(section 2.2.2.3\)](#) message needs to be upgraded.

The client MUST decrypt (see section [5.1.5](#)) the **EncryptedLicenseInfo** field of the Server Upgrade License message using the **license encryption key** to get a [New License Information \(section 2.2.2.6.1\)](#) structure.

It then MUST generate a **MAC** checksum over the decrypted New License Information structure and compares it with the received checksum to verify the data integrity.

The client then MAY store the binary data for the CAL received in the **pbLicenseInfo** field of the New License Information structure in the client's **license store**, using the **dwVersion**, **pbScope**, **pbCompanyName**, and **pbProductId** fields of this structure as indexing information. This binary data replaces the binary data of any previously held CAL.

3.3.5.7 Processing Server New Licenses

The client MUST receive a new license in the [Server New License](#) message. This message is processed in the same way that the client processes the [Server Upgrade License](#) message (see [Processing Server Upgrade Licenses](#)).

3.3.5.8 Handling Out-of-Sequence or Unrecognized Messages

If the client receives a message that is not expected according to the [Licensing PDU Flow](#), or a malformed or an unrecognized message, the client MUST disconnect the RDP connection.

3.3.5.9 Handling Invalid MACs

If the **MAC** generated over decrypted fields of a message does not match the MAC contained in the message, the client MAY send a [License Error message](#) with an error code of ERR_INVALID_MAC and a state transition code of ST_TOTAL_ABORT. The client then MUST disconnect the RDP connection.

3.3.6 Timer Events

None.

3.3.7 Other Local Events

None.


```

000007E0 43 01 4e 4e 6f 56 ca e0-ee d0 f9 4e a6 62 63 ff C.NNoV.....N.bc.
000007F0 da 0b c9 15 61 6c ed 6b-0b c4 58 53 86 0f 8c 0c ....al.k..XS....
00000800 1a 2e df c1 f2 43 48 d4-af 0a 78 36 b2 51 32 28 .....CH...x6.Q2(
00000810 6c c2 75 79 3f 6e 99 66-88 3e 34 d3 7f 6d 9d 07 l.uy?n.f.>4..m..
00000820 e4 6b eb 84 e2 0a bb ca-7d 3a 40 71 b0 be 47 9f .k.....):@q..G.
00000830 12 58 31 61 2b 9b 4a 9a-49 8f e5 b4 0c f5 04 4d .Xla+.J.I.....M
00000840 3c ce bc d2 79 15 d9 28-f4 23 56 77 9f 38 64 3e <...y..(.#Vw.8d>
00000850 03 88 92 04 26 76 b9 b5-df 19 d0 78 4b 7a 60 40 ....&v.....xKz`@
00000860 23 91 f1 15 22 2b b4 e7-02 54 a9 16 21 5b 60 96 #..." +...T..![`.
00000870 a9 5c 00 00 00 00 00 00-00 00 00 00 00 00 00 .\.....
00000880 00 00 01 00 00 00 0e 00-0e 00 6d 69 63 72 6f 73 .....micros
00000890 6f 66 74 2e 63 6f 6d 00 oft.com.

```

```

0x00: LICENSE_PREAMBLE (4 Bytes)
01 -> LICENSE_PREAMBLE::bMsgType = LICENSE_REQUEST (1 Byte)

03 -> LICENSE_PREAMBLE::bVersion = 3 (RDP 5.0, 5.2, 6.0) (1 Byte)

98 -\
08 -/ LICENSE_PREAMBLE::wMsgSize = 0x898 (2 Bytes)

0x04: ServerRandom (0x20 Bytes)
84 ef ae 20 b1 d5 9e 36 -\
49 1a e8 2e 0a 99 89 ac -|
49 a6 47 4f-33 9b 5a b9 -|
95 03 a6 c6 c2 3c 3f 61 -/ SERVER_RANDOM

0x24: Product Info (4 + 4 + 0x2c + 4 + 8 = 0x40 bytes)
00 -\
00 -|
06 -|
00 -/ PRODUCT_INFO::dwVersion = 0x00060000

2c -\
00 -|
00 -|
00 -/ PRODUCT_INFO::cbCompanyName = 0x2c (44 bytes)

4d 00 69 00 63 00 72 00 -\
6f 00 73 00 6f 00 66 00 -|
74 00 20 00 43 00 6f 00 -|
72 00 70 00 6f 00 72 00 -|
61 00 74 00 69 00 6f 00 -|
6e 00 00 00 -/ PRODUCT_INFO::pbCompanyName

08 -\
00 -|
00 -|
00 -/ PRODUCT_INFO::cbProductId = 0x8 (8 bytes)

41 00 30 00 32 00 00 00 -> PRODUCT_INFO::pbProductId
0x64: KeyExchangeList (2 + 2 + 4 = 8 bytes)
0d -\
00 -/ KeyExchangeList::wBlobType = BB_KEY_EXCHG_ALG_BLOB

04 -\
00 -/ KeyExchangeList::wBlobLen = 4

01 -\
00 -|
00 -|
00 -/ KeyExchangeList::pBlob

0x6c: Server Certificate (2 + 2 + 0x812 = 0x816 bytes)
03 -\
00 -/ ServerCertificate::wBlobType = BB_CERTIFICATE_BLOB

12 -\
08 -/ ServerCertificate::wBlobLen = 0x812 bytes

```

```

02 00 00 80-02 00 00 00 f5 02 00 00 -\
30 82 02 f1 30 82 01 dd-a0 03 02 01 02 02 08 01 -|
9e 24 a2 f2 ae 90 80 30-09 06 05 2b 0e 03 02 1d -|
05 00 30 32 31 30 30 13-06 03 55 04 03 1e 0c 00 -|
52 00 4f 00 44 00 45 00-4e 00 54 30 19 06 03 55 -|
04 07 1e 12 00 57 00 4f-00 52 00 4b 00 47 00 52 -|
00 4f 00 55 00 50 30 1e-17 0d 37 30 30 35 32 37 -|
30 31 31 31 30 33 5a 17-0d 34 39 30 35 32 37 30 -|
31 31 31 30 33 5a 30 32-31 30 30 13 06 03 55 04 -|
03 1e 0c 00 52 00 4f 00-44 00 45 00 4e 00 54 30 -|
19 06 03 55 04 07 1e 12-00 57 00 4f 00 52 00 4b -|
00 47 00 52 00 4f 00 55-00 50 30 82 01 22 30 0d -|
06 09 2a 86 48 86 f7 0d-01 01 01 05 00 03 82 01 -|
0f 00 30 82 01 0a 02 82-01 01 00 88 ad 7c 8f 8b -|
82 76 5a bd 8f 6f 62 18-e1 d9 aa 41 fd ed 68 01 -|
c6 34 35 b0 29 04 ca 4a-4a 1c 7e 80 14 f7 8e 77 -|
b8 25 ff 16 47 6f bd e2-34 3d 2e 02 b9 53 e4 33 -|
75 ad 73 28 80 a0 4d fc-6c c0 22 53 1b 2c f8 f5 -|
01 60 19 7e 79 19 39 8d-b5 ce 39 58 dd 55 24 3b -|
55 7b 43 c1 7f 14 2f b0-64 3a 54 95 2b 88 49 0c -|
61 2d ac f8 45 f5 da 88-18 5f ae 42 f8 75 c7 26 -|
6d b5 bb 39 6f cc 55 1b-32 11 38 8d e4 e9 44 84 -|
11 36 a2 61 76 aa 4c b4-e3 55 0f e4 77 8e de e3 -|
a9 ea b7 41 94 00 58 aa-c9 34 a2 98 c6 01 1a 76 -|
14 01 a8 dc 30 7c 77 5a-20 71 5a a2 3f af 13 7e -|
e8 fd 84 a2 5b cf 25 e9-c7 8f a8 f2 8b 84 c7 04 -|
5e 53 73 4e 0e 89 a3 3c-e7 68 5c 24 b7 80 53 3c -|
54 c8 c1 53 aa 71 71 3d-36 15 d6 6a 9d 7d de ae -|
f9 e6 af 57 ae b9 01 96-5d e0 4d cd ed c8 d7 f3 -|
01 03 38 10 be 7c 42 67-01 a7 23 02 03 01 00 01 -|
a3 13 30 11 30 0f 06 03-55 1d 13 04 08 30 06 01 -|
01 ff 02 01 00 30 09 06-05 2b 0e 03 02 1d 05 00 -|
03 82 01 01 00 81 dd d2-d3 33 d4 a3 b6 8e 6e 7d -|
9f fd 73 9f 31 0b dd 42-82 3f 7e 21 df 28 cc 59 -|
ca 6a c0 a9 3d 30 7d e1-91 db 77 6b 8b 10 e6 fd -|
bc 3c a3 58 48 c2 36 dd-a0 0b f5 8e 13 da 7b 04 -|
08 44 b4 f2 a8 0d 1e 0b-1d 1a 3f f9 9b 4b 5a 54 -|
c5 b3 b4 03 93 75 b3 72-5c 3d cf 63 0f 15 e1 64 -|
58 de 52 8d 97 79 0e a4-34 d5 66 05 58 b8 6e 79 -|
b2 09 86 d5 f0 ed c4 6b-4c ab 02 b8 16 5f 3b ed -|
88 5f d1 de 44 e3 73 47-21 f7 03 ce e1 6d 10 0f -|
95 cf 7c a2 7a a6 bf 20-db e1 93 04 c8 5e 6a be -|
c8 01 5d 27 b2 03 0f 66-75 e7 cb ea 8d 4e 98 9d -|
22 ed 28 40 d2 7d a4 4b-ef cc bf 01 2a 6d 3a 3e -|
be 47 38 f8 ea a4 c6 30-1d 5e 25 cf fb e8 3d 42 -|
dd 29 e8 99 89 9e bf 39-ee 77 09 d9 3e 8b 52 36 -|
b6 bb 8b bd 0d b2 52 aa-2c cf 38 4e 4d cf 1d 6d -|
5d 25 17 ac 2c f6 f0 65-5a c9 fe 31 53 b4 f0 0c -|
94 4e 0d 54 8e fd 04 00-00 30 82 04 f9 30 82 03 -|
e5 a0 03 02 01 02 02 05-01 00 00 00 02 30 09 06 -|
05 2b 0e 03 02 1d 05 00-30 32 31 30 30 13 06 03 -|
55 04 03 1e 0c 00 52 00-4f 00 44 00 45 00 4e 00 -|
54 30 19 06 03 55 04 07-1e 12 00 57 00 4f 00 52 -|
00 4b 00 47 00 52 00 4f-00 55 00 50 30 1e 17 0d -|
30 37 30 35 32 36 31 32-34 35 35 33 5a 17 0d 33 -|
38 30 31 31 39 30 33 31-34 30 37 5a 30 81 92 31 -|
81 8f 30 23 06 03 55 04-03 1e 1c 00 6e 00 63 00 -|
61 00 6c 00 72 00 70 00-63 00 3a 00 52 00 4f 00 -|
44 00 45 00 4e 00 54 30-23 06 03 55 04 07 1e 1c -|
00 6e 00 63 00 61 00 6c-00 72 00 70 00 63 00 3a -|
00 52 00 4f 00 44 00 45-00 4e 00 54 30 43 06 03 -|
55 04 05 1e 3c 00 31 00-42 00 63 00 4b 00 65 00 -|
62 00 68 00 70 00 58 00-5a 00 74 00 4c 00 71 00 -|
4f 00 37 00 53 00 51 00-6e 00 42 00 70 00 52 00 -|
66 00 75 00 64 00 64 00-64 00 59 00 3d 00 0d 00 -|
0a 30 82 01 1e 30 09 06-05 2b 0e 03 02 0f 05 00 -|
03 82 01 0f 00 30 82 01-0a 02 82 01 01 00 c8 90 -|

```

```

6b f0 c6 58 81 a6 89 1c-0e f2 f6 d9 82 12 71 a5 -|
6e 51 db e0 32 66 aa 91-77 0e 88 ab 44 b7 d3 97 -|
da 78 8f 0e 44 26 46 7f-16 d4 c6 63 eb ca 55 e5 -|
4e 8b 2d a6 6d 83 95 a7-a8 6a fa d0 be 26 80 ae -|
ab 0a 64 90 32 8c df 5c-f8 f9 d0 7e d1 6b 3a 29 -|
7e 7d bd 02 a3 86 6c fd-a5 35 71 da 21 b4 ee a4 -|
97 f3 a8 b2 12 db a4 27-57 36 c9 08 22 5c 54 f7 -|
99 7b a3 2f b8 5c d5 16-b8 19 27 6b 71 97 14 5b -|
e8 1f 23 e8 5c b8 1b 73-4b 6e 7a 03 13 ff 97 e9 -|
62 b9 4a a0 51 23 c3 6c-32 3e 02 f2 63 97 23 1c -|
c5 78 d8 fc b7 07 4b b0-56 0f 74 df c5 56 28 e4 -|
96 fd 20 8e 65 5a e6 45-ed c1 05 3e ab 58 55 40 -|
af e2 47 a0 4c 49 a3 8d-39 e3 66 5f 93 33 6d f8 -|
5f c5 54 e5 fb 57 3a de-45 12 b5 c7 05 4b 88 1f -|
b4 35 0f 7c c0 75 17 c6-67 dd 48 80 cb 0a be 9d -|
f6 93 60 65 34 eb 97 af-65 6d df bf 6f 5b 02 03 -|
01 00 01 a3 82 01 bf 30-82 01 bb 30 14 06 09 2b -|
06 01 04 01 82 37 12 04-01 01 ff 04 04 01 00 05 -|
00 30 3c 06 09 2b 06 01-04 01 82 37 12 02 01 01 -|
ff 04 2c 4d 00 69 00 63-00 72 00 6f 00 73 00 6f -|
00 66 00 74 00 20 00 43-00 6f 00 72 00 70 00 6f -|
00 72 00 61 00 74 00 69-00 6f 00 6e 00 00 00 30 -|
81 cd 06 09 2b 06 01 04-01 82 37 12 05 01 01 ff -|
04 81 bc 00 30 00 00 01-00 00 00 02 00 00 00 09 -|
04 00 00 1c 00 4a 00 66-00 4a 00 b0 00 01 00 33 -|
00 64 00 32 00 36 00 37-00 39 00 35 00 34 00 2d -|
00 65 00 65 00 62 00 37-00 2d 00 31 00 31 00 64 -|
00 31 00 2d 00 62 00 39-00 34 00 65 00 2d 00 30 -|
00 30 00 63 00 30 00 34-00 66 00 61 00 33 00 30 -|
00 38 00 30 00 64 00 00-00 33 00 64 00 32 00 36 -|
00 37 00 39 00 35 00 34-00 2d 00 65 00 65 00 62 -|
00 37 00 2d 00 31 00 31-00 64 00 31 00 2d 00 62 -|
00 39 00 34 00 65 00 2d-00 30 00 30 00 63 00 30 -|
00 34 00 66 00 61 00 33-00 30 00 38 00 30 00 64 -|
00 00 00 00 00 00 10 00-80 64 00 00 00 00 00 30 -|
6e 06 09 2b 06 01 04 01-82 37 12 06 01 01 ff 04 -|
5e 00 30 00 00 00 00 0e-00 3e 00 52 00 4f 00 44 -|
00 45 00 4e 00 54 00 00-00 37 00 38 00 34 00 34 -|
00 30 00 2d 00 30 00 30-00 36 00 2d 00 35 00 38 -|
00 36 00 37 00 30 00 34-00 35 00 2d 00 37 00 30 -|
00 33 00 34 00 37 00 00-00 57 00 4f 00 52 00 4b -|
00 47 00 52 00 4f 00 55-00 50 00 00 00 00 00 30 -|
25 06 03 55 1d 23 01 01-ff 04 1b 30 19 a1 10 a4 -|
0e 52 00 4f 00 44 00 45-00 4e 00 54 00 00 00 82 -|
05 01 00 00 00 02 30 09-06 05 2b 0e 03 02 1d 05 -|
00 03 82 01 01 00 2e eb-c7 0d b8 1d 47 11 9d 09 -|
88 9b 51 dc 45 dd 56 51-e2 d1 23 11 39 9b 2d da -|
c7 fe 7a d7 84 e3 3d 54-77 97 4d 19 92 30 64 a0 -|
47 c6 2f 6d 93 d2 64 7c-76 c8 26 45 ad 5a 44 54 -|
ea f6 4b 28 77 1f 77 ea-ec 74 02 38 68 9e 79 14 -|
72 83 34 74 62 d2 c1 0c-a4 0b f2 a9 b0 38 bb 7c -|
d0 ae be bf 74 47 16 a0-a2 d3 fc 1d b9 ba 26 10 -|
06 ef ba 1d 43 01 4e 4e-6f 56 ca e0 ee d0 f9 4e -|
a6 62 63 ff da 0b c9 15-61 6c ed 6b 0b c4 58 53 -|
86 0f 8c 0c 1a 2e df c1-f2 43 48 d4 af 0a 78 36 -|
b2 51 32 28 6c c2 75 79-3f 6e 99 66 88 3e 34 d3 -|
7f 6d 9d 07 e4 6b eb 84-e2 0a bb ca 7d 3a 40 71 -|
b0 be 47 9f 12 58 31 61-2b 9b 4a 9a 49 8f e5 b4 -|
0c f5 04 4d 3c ce bc d2-79 15 d9 28 f4 23 56 77 -|
9f 38 64 3e 03 88 92 04-26 76 b9 b5 df 19 d0 78 -|
4b 7a 60 40 23 91 f1 15-22 2b b4 e7 02 54 a9 16 -|
21 5b 60 96 a9 5c 00 00-00 00 00 00 00 00 00 -|
00 00 00 00 00 00 -|
ServerCertificate::pBlob

```

0x882: ScopeList

```

01 -\
00 -|

```

```

00 -|
00 -/ ScopeList::ScopeCount

0e 00 0e 00 6d 69 63 72 -\
6f 73 6f 66 74 2e 63 6f -|
6d 00 -/ ScopeList::ScopeArray

```

4.2 CLIENT NEW LICENSE REQUEST

If the client does not already have a license in its store, it sends the [Client New License Request](#) message as a response to the [Server License Request](#) message. See sections 2.2.2.2 and [3.3.5.2](#). For information on ExtendedError, see [\[MS-EERR\]](#).

```

00000000: 13 83 55 01 01 00 00 00-00 00 01 04 dc 73 a0 c8 ..U.....s..
00000010: 69 25 6b 18 af 0b 94 7a-a9 a5 20 af 8b bc 0d cc i%k....z.. ....
00000020: a3 95 b7 b9 eb 81 5d be-0a 10 9c d8 02 00 08 01 .....].....
00000030: da 9c 5d a6 68 9d a3 90-67 24 f3 3a ea a1 e2 68 ..].h...g$.....h
00000040: ad 12 f5 f6 0b 7a ac 92-b1 69 6f 42 55 8a a0 e2 .....z....ioBU...
00000050: 9b 2c d0 c7 ee 33 6c 47-79 c3 1e bf 03 8b 95 70 ,...3lGy.....p
00000060: 07 a2 be ee 54 02 68 f8-90 d7 fe 2c 08 e1 6b 2d ....T.h....,..k-
00000070: ff 94 76 72 5f 7a 76 75-32 55 cc 58 61 63 a5 64 ..vr zvu2U.Xac.d
00000080: f1 6e c3 07 81 82 6f 88-73 62 fc 28 65 91 c2 c8 .n....o.sb.(e...
00000090: 9f 05 b0 d3 93 12 bf 6a-50 18 99 2d 4d c4 7f 74 .....jP..-M..t
000000A0: d3 30 9f 16 78 a5 df aa-83 65 4f 77 30 42 e0 d7 .0..x....eOw0B..
000000B0: 69 c8 4d a5 73 11 59 35-b9 a7 e2 b0 f6 e3 b9 39 i.M.s.Y5.....9
000000C0: c3 d4 e4 6b ca 40 9a ac-66 e6 1a a4 1b 39 7e 09 ...k.@..f....9~.
000000D0: e3 72 99 dd 90 62 55 97-a9 04 c7 51 aa a2 01 cb .r...bU....Q....
000000E0: 5a 63 4d 1a e5 99 c3 b1-2a 73 e8 9a 00 46 92 59 ZcM.....*s...F.Y
000000F0: 39 a3 80 a1 ac 90 52 ea-63 81 49 7d f3 2d 5c c3 9.....R.c.I}.-\
00000100: 19 9f ed fe 81 1d 8c 04-1c d9 23 d2 6d 80 84 f3 .....#.m...
00000110: 00 f2 b1 69 2f cd b3 9f-69 ee 60 3e 4b b5 be 5a ...i/...i.`>K..Z
00000120: 09 83 0b bc 3d 3e 05 47-65 96 31 8c 6b c5 e6 a0 ....=>.Ge.1.k...
00000130: 00 00 00 00 00 00 00-0f 00 0e 00 41 64 6d 69 .....Admi
00000140: 6e 69 73 74 72 61 74 6f-72 00 10 00 07 00 52 4f nistrator....RO
00000150: 44 45 4e 54 00 DENT.

```

```

0x00: LICENSE_PREAMBLE (4 bytes)
13 -> LICENSE_PREAMBLE::bMsgType = NEW LICENSE REQUEST

83 -> LICENSE_PREAMBLE::bVersion = 0x80
      (ExtendedError supported) | 0x3 (RDP 5.0,5.2,6.0)

55 -\
01 -/ LICENSE_PREAMBLE::wMsgSize = 0x155
0x04: PreferredKeyGenAlgo (4 bytes)
01 -\
00 -|
00 -|
00 -/ PreferredKeyGenAlgo = 0x01
0x08: PlatformId (4 bytes)
00 -\
00 -|
01 -|
04 -/ PlatformId = CLIENT OS ID WINNT POST 52 |
      CLIENT__IMAGE_ID_MICROSOFT = 0x04010000
0x0c: ClientRandom (0x20 bytes)
dc 73 a0 c8 69 25 6b 18
af 0b 94 7a a9 a5 20 af
8b bc 0d cc a3 95 b7 b9
eb 81 5d be 0a 10 9c d8
0x2c: EncryptedPreMaster (2 + 2 + 0x108 = 0x10c bytes)
02 -\
00 -/ EncryptedPreMaster::wBlobType

08 -\

```

```

01 -/ EncryptedPreMaster::wBlobLen = 0x108 bytes

da 9c 5d a6 68 9d a3 90-67 24 f3 3a ea a1 e2 68 -\
ad 12 f5 f6 0b 7a ac 92-b1 69 6f 42 55 8a a0 e2 -|
9b 2c d0 c7 ee 33 6c 47-79 c3 1e bf 03 8b 95 70 -|
07 a2 be ee 54 02 68 f8-90 d7 fe 2c 08 e1 6b 2d -|
ff 94 76 72 5f 7a 76 75-32 55 cc 58 61 63 a5 64 -|
f1 6e c3 07 81 82 6f 88-73 62 fc 28 65 91 c2 c8 -|
9f 05 b0 d3 93 12 bf 6a-50 18 99 2d 4d c4 7f 74 -|
d3 30 9f 16 78 a5 df aa-83 65 4f 77 30 42 e0 d7 -|
69 c8 4d a5 73 11 59 35-b9 a7 e2 b0 f6 e3 b9 39 -|
c3 d4 e4 6b ca 40 9a ac-66 e6 1a a4 1b 39 7e 09 -|
e3 72 99 dd 90 62 55 97-a9 04 c7 51 aa a2 01 cb -|
5a 63 4d 1a e5 99 c3 b1-2a 73 e8 9a 00 46 92 59 -|
39 a3 80 a1 ac 90 52 ea-63 81 49 7d f3 2d 5c c3 -|
19 9f ed fe 81 1d 8c 04-1c d9 23 d2 6d 80 84 f3 -|
00 f2 b1 69 2f cd b3 9f-69 ee 60 3e 4b b5 be 5a -|
09 83 0b bc 3d 3e 05 47-65 96 31 8c 6b c5 e6 a0 -|
00 00 00 00 00 00 00 00 -/
EncryptedPreMaster::pBlob

The decrypted pre-master data for the above blob is as
cf 7a db cb fb 0e 15 23-87 1c 84 81 ba 9d 4e 15
bb d2 56 bd d8 f7 f3 16-cc 35 3b e1 93 42 78 dd
92 9a e4 7a e2 99 d4 73-b1 aa 6f 55 94 3b c9 bc
0x138: ClientUserName (2 + 2 + 0xe = 0x12 bytes)
0f -\
00 -/ ClientUserName::wBlobType = BB_CLIENT_USER_NAME

0e -\
00 -/ ClientUserName::wBlobLen = 0xe bytes

41 64 6d 69 6e 69 73 74 -\
72 61 74 6f 72 00 -/ ClientUserName::pBlob
0x14a: ClientMachineName (2 + 2 + 7 = 0xb bytes)
10 -\
00 -/ ClientMachineName::wBlobType = BB_CLIENT_MACHINE_NAME

07 -\
00 -/ ClientMachineName::wBlobLen = 7 bytes

52 4f 44 45 4e 54 00 -> ClientMachineName::pBlob

```

4.3 CLIENT LICENSE INFO

If the client already has a license in its **store**, it sends the license in the [Client License Information](#) message as a response to the [Server License Request](#) message. See sections 2.2.2.3 and [3.3.5.3](#).

```

00000000: 12 83 fd 08 01 00 00 00-00 00 01 04 26 c9 32 34 .....&.24
00000010: 7d 2b e1 75 50 5e 47 7e-76 8d 78 7b bb 21 aa b7 }+.uP^G~v.x{!..
00000020: b0 b8 ea 6c dd c1 b0 01-e6 13 be d8 02 00 08 01 ...l.....
00000030: b1 fa 1c 25 d9 5e 9d 04-cd d2 d4 34 c6 a2 e6 f3 ...%.^.....4....
00000040: a2 bf bc 53 8a 0e 15 cf-1c 1a 99 bb 49 dc 9e 71 ...S.....I..q
00000050: 39 03 78 ce 5d 72 29 09-3a 86 b1 1d e8 b1 38 07 9.x.]r)..8.
00000060: 29 62 87 84 1e cc 95 b0-49 19 a0 cf 27 90 9a ef )b.....I...'...
00000070: d6 a9 9a f4 66 d9 d2 e9-64 ee 4a aa e4 22 d6 44 ....f...d.J..".D
00000080: b8 72 79 3a d2 82 09 e1-1f 85 69 3e 09 68 c3 fa .ry:.....i>.h..
00000090: 36 e2 ba 3c c5 4c 46 7d-68 42 2d 03 bf e0 03 d4 6...<.LF}hB-....
000000A0: 13 45 da 21 14 32 41 86-c4 00 d0 42 ef 68 ee 9c .E.!2A....B.h..
000000B0: 0a 1a d9 42 6f 64 7b e4-b9 61 db be 52 e6 3e 63 ...Bod{...a..R.>c
000000C0: dc ec 7e 97 a0 0d 87 82-be 56 be 29 8b 7d 27 bd ..~.....V..)''.
000000D0: 2a 31 57 b0 3a 3a ad 35-70 d4 bd 29 20 8d 82 dc *1W...:5p..) ...
000000E0: f5 7c f9 3a 2f 0d a2 5b-55 7a 95 37 af d5 d8 47 .|.:/..[Uz.7...G
000000F0: f9 1d a3 89 5a cf 66 b2-4d b2 83 db c2 45 c8 3b ....Z.f.M....E.;
00000100: 86 31 1c 2f a7 59 1a 09-89 56 ed 18 09 4c 35 1a .1./..Y...V....L5.

```



```

01 -\
00 -|
00 -|
00 -/ CLIENT_LICENSE_INFO::dwPrefKeyExchangeAlg = 1

0x08: PlatformId (4 bytes)
00 -\
00 -|
01 -|
04 -/ CLIENT_LICENSE_INFO::dwPlatformId = 0x04010000
0x0c: Client Random (0x20 bytes)
      26 c9 32 34 7d 2b e1 75 -\
50 5e 47 7e-76 8d 78 7b -|
bb 21 aa b7 b0 b8 ea 6c -|
dd c1 b0 01-e6 13 be d8 -/ CLIENT_LICENSE_INFO::ClientRandom
0x2c: EncryptedPreMaster (2 + 2 + 0x108 = 0x10c bytes)
02 -\
00 -/ EncryptedPreMasterSecret::wBlobType

08 -\
01 -/ EncryptedPreMasterSecret::wBlobLen

b1 fa 1c 25 d9 5e 9d 04-cd d2 d4 34 c6 a2 e6 f3 -\
a2 bf bc 53 8a 0e 15 cf-1c 1a 99 bb 49 dc 9e 71 -|
39 03 78 ce 5d 72 29 09-3a 86 b1 1d e8 b1 38 07 -|
29 62 87 84 1e cc 95 b0-49 19 a0 cf 27 90 9a ef -|
d6 a9 9a f4 66 d9 d2 e9-64 ee 4a aa e4 22 d6 44 -|
b8 72 79 3a d2 82 09 e1-1f 85 69 3e 09 68 c3 fa -|
36 e2 ba 3c c5 4c 46 7d-68 42 2d 03 bf e0 03 d4 -|
13 45 da 21 14 32 41 86-c4 00 d0 42 ef 68 ee 9c -|
0a 1a d9 42 6f 64 7b e4-b9 61 db be 52 e6 3e 63 -|
dc ec 7e 97 a0 0d 87 82-be 56 be 29 8b 7d 27 bd -|
2a 31 57 b0 3a 3a ad 35-70 d4 bd 29 20 8d 82 dc -|
f5 7c f9 3a 2f 0d a2 5b-55 7a 95 37 af d5 d8 47 -|
f9 1d a3 89 5a cf 66 b2-4d b2 83 db c2 45 c8 3b -|
86 31 1c 2f a7 59 1a 09-89 56 ed 18 09 4c 35 1a -|
47 5c b3 35 f2 09 01 bf-6a 02 bc fc be 75 a6 8c -|
70 2e 3b 03 73 64 b4 13-89 0c 1e a4 3e 49 e9 b9 -|
00 00 00 00 00 00 00 00 -/
EncryptedPreMasterSecret::pBlob
0x138: LicenseInfo (2 + 2 + 0x0799 = 0x79d bytes)
01 -\
00 -/ LicenseInfo::wBlobType = BB_DATA_BLOB

99 -\
07 -/ LicenseInfo::wBlobLen = 0x799

      30 82 07 95 -\
06 09 2a 86 48 86 f7 0d-01 07 02 a0 82 07 86 30 -|
82 07 82 02 01 01 31 00-30 0b 06 09 2a 86 48 86 -|
f7 0d 01 07 01 a0 82 07-6a 30 82 02 f1 30 82 01 -|
dd a0 03 02 01 02 02 08-01 9e 27 4d 68 ac ed 20 -|
30 09 06 05 2b 0e 03 02-1d 05 00 30 32 31 30 30 -|
13 06 03 55 04 03 1e 0c-00 52 00 4f 00 44 00 45 -|
00 4e 00 54 30 19 06 03-55 04 07 1e 12 00 57 00 -|
4f 00 52 00 4b 00 47 00-52 00 4f 00 55 00 50 30
1e 17 0d 37 30 30 35 33-30 31 30 33 36 31 38 5a -|
17 0d 34 39 30 35 33 30-31 30 33 36 31 38 5a 30 -|
32 31 30 30 13 06 03 55-04 03 1e 0c 00 52 00 4f -|
00 44 00 45 00 4e 00 54-30 19 06 03 55 04 07 1e -|
12 00 57 00 4f 00 52 00-4b 00 47 00 52 00 4f 00 -|
55 00 50 30 82 01 22 30-0d 06 09 2a 86 48 86 f7 -|
0d 01 01 01 05 00 03 82-01 0f 00 30 82 01 0a 02 -|
82 01 01 00 88 ad 7c 8f-8b 82 76 5a bd 8f 6f 62 -|
18 e1 d9 aa 41 fd ed 68-01 c6 34 35 b0 29 04 ca -|
4a 4a 1c 7e 80 14 f7 8e-77 b8 25 ff 16 47 6f bd -|
e2 34 3d 2e 02 b9 53 e4-33 75 ad 73 28 80 a0 4d -|
fc 6c c0 22 53 1b 2c f8-f5 01 60 19 7e 79 19 39 -|
8d b5 ce 39 58 dd 55 24-3b 55 7b 43 c1 7f 14 2f -|

```

b0 64 3a 54 95 2b 88 49-0c 61 2d ac f8 45 f5 da -|
88 18 5f ae 42 f8 75 c7-26 6d b5 bb 39 6f cc 55 -|
1b 32 11 38 8d e4 e9 44-84 11 36 a2 61 76 aa 4c -|
b4 e3 55 0f e4 77 8e de-e3 a9 ea b7 41 94 00 58 -|
aa c9 34 a2 98 c6 01 1a-76 14 01 a8 dc 30 7c 77 -|
5a 20 71 5a a2 3f af 13-7e e8 fd 84 a2 5b cf 25 -|
e9 c7 8f a8 f2 8b 84 c7-04 5e 53 73 4e 0e 89 a3 -|
3c e7 68 5c 24 b7 80 53-3c 54 c8 c1 53 aa 71 71 -|
3d 36 15 d6 6a 9d 7d de-ae f9 e6 af 57 ae b9 01 -|
96 5d e0 4d cd ed c8 d7-f3 01 03 38 10 be 7c 42 -|
67 01 a7 23 02 03 01 00-01 a3 13 30 11 30 0f 06 -|
03 55 1d 13 04 08 30 06-01 01 ff 02 01 00 30 09 -|
06 05 2b 0e 03 02 1d 05-00 03 82 01 01 00 70 db -|
21 2b 84 9a 7a c3 b1 68-fa c0 00 8b 71 ab 43 9f -|
b6 7b b7 1f 20 83 ac 0a-b5 0e ad b6 36 ef 65 17 -|
99 86 8a 3d ba 0c 53 2e-a3 75 a0 f3 11 3d e7 65 -|
4b ae 3c 42 70 11 dc ca-83 c0 be 3e 97 71 84 69 -|
d6 a8 27 33 9b 3e 17 3c-a0 4c 64 ca 20 37 a4 11 -|
a9 28 8f b7 18 96 69 15-0d 74 04 75 2a 00 c7 a6 -|
6a be ac b3 f2 fb 06 1b-6c 11 bd 96 e2 34 74 5d -|
f5 98 8f 3a 8d 69 08 6f-53 12 4e 39 80 90 ce 8b -|
5e 88 23 2d fd 55 fd 58-3d 39 27 b3 7c 57 fe 3b -|
ab 62 26 60 e2 d0 c8 f4-02 23 16 c3 52 5d 9f 05 -|
49 a2 71 2d 6d 5b 90 dd-bf e5 a9 2e f1 85 8a 8a -|
b8 a9 6b 13 cc 8d 4c 22-41 ad 32 1e 3b 4b 89 37 -|
66 df 1e a5 4a 03 52 1c-d9 19 79 22 d4 a7 3b 47 -|
93 a9 0c 03 6a d8 5f fc-c0 75 33 e5 26 da f7 4a -|
77 d8 f1 30 80 39 38 1e-86 1d 97 00 9c 0e ba 00 -|
54 8a c0 12 32 6f 3d c4-15 f9 50 f8 ce 95 30 82 -|
04 71 30 82 03 5d a0 03-02 01 02 02 05 03 00 00 -|
00 0f 30 09 06 05 2b 0e-03 02 1d 05 00 30 32 31 -|
30 30 13 06 03 55 04 03-1e 0c 00 52 00 4f 00 44 -|
00 45 00 4e 00 54 30 19-06 03 55 04 07 1e 12 00 -|
57 00 4f 00 52 00 4b 00-47 00 52 00 4f 00 55 00 -|
50 30 1e 17 0d 30 37 30-36 32 30 31 34 35 31 33 -|
35 5a 17 0d 30 37 30 39-31 38 31 34 35 31 33 35 -|
5a 30 7f 31 7d 30 13 06-03 55 04 03 1e 0c 00 52 -|
00 4f 00 44 00 45 00 4e-00 54 30 21 06 03 55 04 -|
07 1e 1a 00 41 00 64 00-6d 00 69 00 6e 00 69 00 -|
73 00 74 00 72 00 61 00-74 00 6f 00 72 30 43 06 -|
03 55 04 05 1e 3c 00 31-00 42 00 63 00 4b 00 65 -|
00 64 00 79 00 32 00 6b-00 72 00 4f 00 34 00 2f -|
00 4d 00 43 00 44 00 4c-00 49 00 31 00 41 00 48 -|
00 5a 00 63 00 50 00 69-00 61 00 73 00 3d 00 0d -|
00 0a 30 82 01 22 30 0d-06 09 2a 86 48 86 f7 0d -|
01 01 01 05 00 03 82 01-0f 00 30 82 01 0a 02 82 -|
01 01 00 88 ad 7c 8f 8b-82 76 5a bd 8f 6f 62 18 -|
e1 d9 aa 41 fd ed 68 01-c6 34 35 b0 29 04 ca 4a -|
4a 1c 7e 80 14 f7 8e 77-b8 25 ff 16 47 6f bd e2 -|
34 3d 2e 02 b9 53 e4 33-75 ad 73 28 80 a0 4d fc -|
6c c0 22 53 1b 2c f8 f5-01 60 19 7e 79 19 39 8d -|
b5 ce 39 58 dd 55 24 3b-55 7b 43 c1 7f 14 2f b0 -|
64 3a 54 95 2b 88 49 0c-61 2d ac f8 45 f5 da 88 -|
18 5f ae 42 f8 75 c7 26-6d b5 bb 39 6f cc 55 1b -|
32 11 38 8d e4 e9 44 84-11 36 a2 61 76 aa 4c b4 -|
e3 55 0f e4 77 8e de e3-a9 ea b7 41 94 00 58 aa -|
c9 34 a2 98 c6 01 1a 76-14 01 a8 dc 30 7c 77 5a -|
20 71 5a a2 3f af 13 7e-e8 fd 84 a2 5b cf 25 e9 -|
c7 8f a8 f2 8b 84 c7 04-5e 53 73 4e 0e 89 a3 3c -|
e7 68 5c 24 b7 80 53 3c-54 c8 c1 53 aa 71 71 3d -|
36 15 d6 6a 9d 7d de ae-f9 e6 af 57 ae b9 01 96 -|
5d e0 4d cd ed c8 d7 f3-01 03 38 10 be 7c 42 67 -|
01 a7 23 02 03 01 00 01-a3 82 01 47 30 82 01 43 -|
30 14 06 09 2b 06 01 04-01 82 37 12 04 01 01 ff -|
04 04 01 00 05 00 30 3c-06 09 2b 06 01 04 01 82 -|
37 12 02 01 01 ff 04 2c-4d 00 69 00 63 00 72 00 -|
6f 00 73 00 6f 00 66 00-74 00 20 00 43 00 6f 00 -|
72 00 70 00 6f 00 72 00-61 00 74 00 69 00 6f 00 -|
6e 00 00 00 30 56 06 09-2b 06 01 04 01 82 37 12 -|

```

05 01 01 ff 04 46 00 30-00 00 01 00 00 00 ff 00 -|
00 00 00 04 00 00 1c 00-08 00 24 00 16 00 3a 00 -|
01 00 41 00 30 00 32 00-00 00 41 00 30 00 32 00 -|
2d 00 36 00 2e 00 30 00-30 00 2d 00 53 00 00 00 -|
06 00 00 00 00 80 64 80-00 00 00 00 30 6e 06 09 -|
2b 06 01 04 01 82 37 12-06 01 01 ff 04 5e 00 30 -|
00 00 00 00 0e 00 3e 00-52 00 4f 00 44 00 45 00 -|
4e 00 54 00 00 00 37 00-38 00 34 00 34 00 30 00 -|
2d 00 30 00 30 00 36 00-2d 00 35 00 38 00 36 00 -|
37 00 30 00 34 00 35 00-2d 00 37 00 30 00 33 00 -|
34 00 37 00 00 00 57 00-4f 00 52 00 4b 00 47 00 -|
52 00 4f 00 55 00 50 00-00 00 00 00 30 25 06 03 -|
55 1d 23 01 01 ff 04 1b-30 19 a1 10 a4 0e 52 00 -|
4f 00 44 00 45 00 4e 00-54 00 00 00 82 05 03 00 -|
00 00 0f 30 09 06 05 2b-0e 03 02 1d 05 00 03 82 -|
01 01 00 13 1b dc 89 d2-fc 54 0c ee 82 45 68 6a -|
72 c3 3e 17 73 96 53 44-39 50 0e 0b 9f 95 d6 2c -|
6b 53 14 9c e5 55 ed 65-df 2a eb 5c 64 85 70 1f -|
bc 96 cf a3 76 b1 72 3b-e1 f6 ad ad ad 2a 14 af -|
ba d0 d6 d5 6d 55 ec 1e-c3 4b ba 06 9c 59 78 93 -|
64 87 4b 03 f9 ee 4c dd-36 5b bd d4 e5 4c 4e da -|
7b c1 ae 23 28 9e 77 6f-0f e6 94 fe 05 22 00 ab -|
63 5b e1 82 45 a6 ec 1f-6f 2c 7b 56 de 78 25 7d -|
10 60 0e 53 42 4b 6c 7a-6b 5d c9 d5 a6 ae c8 c8 -|
52 29 d6 42 56 02 ec f9-23 a8 8c 8d 89 c9 7c 84 -|
07 fc 33 e1 1e ea e2 8f-2b be 8f a9 d3 d1 e1 5e -|
0b dc b6 43 6e 33 0a f4-2e 9d 0c c9 58 54 34 aa -|
e1 d2 a2 e4 90 02 23 26-a0 92 26 26 0a 83 b4 4d -|
d9 4b ef eb 9d a9 24 3f-92 8b db 04 7b 9d 64 91 -|
a4 4b d2 6e 51 05 08 c9-91 af 31 26 55 21 b1 ea -|
ce a3 a4 0d 5e 4c 46 db-16 2d 98 dc 60 19 b8 1b -|
b9 cd fb 31 00 -/
0x8d5: EncryptedHWID (2 + 2 + 0x14 = 0x18 bytes)
01 -\
00 -/ EncryptedHWID::wBlobType

14 -\
00 -/ EncryptedHWID::wBlobLen

b9 30 59 3b 93 61 c9 f6 -\
b6 0b 1f dc 1a 85 67 39 -|
dc 29 65 62 -/ EncryptedHWID::pBlob

0x:8ed: MACData (0x10 bytes)
42 a2 13 c7 54 ae b5 d5 -\
24 66 54 f3 1b af 8d fb -/ MACData

```

4.4 SERVER PLATFORM CHALLENGE

The server sends the [platform challenge](#) to the client to authenticate the client. See sections [2.2.2.4](#) and [3.2.5.4](#).

```

00000000: 02 03 26 00 ff ff ff ff-50 f7 0a 00 46 37 85 54 ..&.....P...F7.T
00000010: 8e c5 91 34 97 5d 78 94-ad 3b 81 da 88 18 56 0f ...4.]x...;....V.
00000020: 3a d1 f1 03 ef 35 :....5

0x00: LICENSE_PREAMBLE (4 bytes)
02 -> LICENSE_PREAMBLE::bMsgType = SERVER_PLATFORM_CHALLENGE

03 -> LICENSE_PREAMBLE::bVersion = 3

26 -\
00 -/ LICENSE_PREAMBLE::wMsgSize
0x04: ConnectFlags (4 bytes)
ff -\

```

```

ff -\
ff -\
ff -/ dwConnectFlags (ignored)
0x08: EncryptedPlatformChallenge (2 + 2 + 0xa = 0xe bytes)
50 -\
f7 -/ EncryptedPlatformChallenge::wBlobType (ignored)

0a -\
00 -/ EncryptedPlatformChallenge::wBlobLen

46 37 85 54 8e c5 91 34 -\
          97 5d -/ EncryptedPlatformChallenge::pBlob

The corresponding decrypted blob for the above is
0x00000000: 54 00 45 00 53 00 54 00 00 00 T.E.S.T...
0x16: MACData
78 94 ad 3b 81 da 88 18 -\
56 0f 3a d1 f1 03 ef 35 -/ MACData

```

4.5 CLIENT PLATFORM CHALLENGE RESPONSE

The client sends the [Client Platform Challenge Response](#) message in response to a [Server Platform Challenge](#) message. The message type in the [licensing preamble](#) is PLATFORM_CHALLENGE_RESPONSE (0x15).

See sections 2.2.2.5 and [3.3.5.5](#) for more information.

```

00000000: 15 83 42 00 01 00 12 00-fa b4 e8 24 cf 56 b2 4e ..B.....$.V.N
00000010: 80 02 bd b6 61 fc df e9-6c 44 01 00 14 00 f8 b5 ....a...lD.....
00000020: e8 25 3d 0f 3f 70 1d da-60 19 16 fe 73 1a 45 7e .%=?p..`...s.E~
00000030: 02 71 38 23 62 5d 10 8b-93 c3 f1 e4 67 1f 4a b6 .q8#b].....g.J.
00000040: 00 0a
..

0x00: LICENSE_PREAMBLE (4 bytes)
15 -> LICENSE_PREAMBLE::bMsgType = CLIENT_PLATFORM_CHALLENGE_REPSONSE

83 -> LICENSE_PREAMBLE::bVersion = 0x80 | 0x3

42 -\
00 -/ LICENSE_PREAMBLE::wMsgSize = 0x42 bytes
0x04: EncryptedPlatformChallengeResponse
(2 + 2 + 0x12 = 0x16 bytes)
01 -\
00 -/ EncryptedPlatformChallengeResponse::wBlobType (ignored)

12 -\
00 -/ EncryptedPlatformChallengeResponse::wBlobLen = 0x12

fa b4 e8 24 cf 56 b2 4e -\
80 02 bd b6 61 fc df e9 -\
6c 44 -/
EncryptedPlatformChallengeResponse::pBlob

The corresponding decrypted platform challenge response data
for the above is
00 -\
01 -/ PLATFORM_CHALLENGE_RESPONSE_DATA::wVersion

00 -\
01 -/ PLATFORM_CHALLENGE_RESPONSE_DATA::wClientType

03 -\
00 -/ PLATFORM_CHALLENGE_RESPONSE_DATA::wLicenseDetailLevel

```

```

0a -\
00 -/ PLATFORM_CHALLENGE_RESPONSE_DATA::cbChallenge

54 -\
00 -|
45 -|
00 -|
53 -|
00 -|
54 -|
00 -|
00 -|
00 -/ PLATFORM_CHALLENGE_RESPONSE_DATA::pbChallenge

0x1a: EncryptedHWID (2 + 2 + 0x14 = 0x18 bytes)
01 -\
00 -/ EncryptedHWID::wBlobType (ignored)

14 -\
00 -/ EncryptedHWID::wBlobLen

f8 b5 e8 25 3d 0f 3f 70 -\
1d da-60 19 16 fe 73 1a -|
45 7e 02 71 -/ EncryptedHWID::pBlob

The corresponding decrypted HWID for the above is
02 -\
00 -|
00 -|
00 -/ CLIENT_HARDWARE_ID::PlatformId

f1 -\
59 -|
87 -|
3e -/ CLIENT_HARDWARE_ID::Data1

c9 -\
d8 -|
98 -|
af -/ CLIENT_HARDWARE_ID::Data2

24 -\
02 -|
f8 -|
f3 -/ CLIENT_HARDWARE_ID::Data3

29 -\
3a -|
f0 -|
26 -/ CLIENT_HARDWARE_ID::Data4

0x32: MACData (0x10 bytes)
38 23 62 5d 10 8b-93 c3 -\
f1 e4 67 1f 4a b6 00 0a -/ MACData

```

4.6 SERVER NEW LICENSE

The server sends the license to the client in the [Server New License](#) message. See sections 2.2.2.7 and [3.2.5.7](#) for more information.

```

00000000: 03 03 07 08 09 00 ef 07-db a3 13 30 79 a3 cd 9e .....0y...
00000010: 48 f4 8f 06 37 1b 45 dd-60 a9 2e 29 26 bf c1 96 H...7.E.^...)&...
00000020: 5e 07 93 9d f2 2d 3e a3-3a ff d5 6d f5 85 30 28 ^....->:...m..0(

```



```

00 -/ EncryptedLicenseInfo::wBlobType = BB_ENCRYPTED_DATA_BLOB

ef -\
07 -/ EncryptedLicenseInfo::wBlobLen = 0x7ef bytes

      db a3 13 30 79 a3 cd 9e -\
48 f4 8f 06 37 1b 45 dd-60 a9 2e 29 26 bf c1 96 -|
5e 07 93 9d f2 2d 3e a3-3a ff d5 6d f5 85 30 28 -|
e1 46 fd 56 d1 20 41 33-94 88 0c 27 23 a0 61 38 -|
60 db 86 d6 ce 2c cd 40-39 55 23 39 12 b9 fd c2 -|
8d 58 0a 37 33 42 5c 61-d7 c8 a0 11 66 e2 45 ba -|
41 39 ea 85 2a 6e 7a b3-e7 27 75 fc 4d c0 fb 0d -|
e8 67 90 b3 3a 40 f0 15-8a 15 8e 2c 99 0f 1c bd -|
d2 08 66 51 9e 6a e6 2c-f7 1f d0 c0 8e 89 76 64 -|
18 58 a1 94 bd ce b1 2d-96 ab 53 cf f8 bf d0 c9 -|
c0 2e e6 a4 0b 50 31 4a-4e d8 47 4b af b8 21 78 -|
bf 09 ac 7f 2d 2d 88 f6-d8 c7 45 33 9f ac 69 f5 -|
88 9d 5c 6e c9 d0 ca 8c-bc a9 d6 07 36 ed 40 95 -|
8a c1 3f 04 41 b3 c9 b3-18 9d 33 1b 04 55 cd 41 -|
df 19 e1 cd a0 a4 35 6e-b7 0a f3 ec 48 10 4f 28 -|
c6 35 f3 9b a2 d5 f7 58-03 4d 9a 16 34 fb 96 0c -|
d5 3a ae 52 1b 2f 1f 1f-31 b2 d9 14 3b 73 0f e3 -|
04 e0 a5 52 89 68 ba 0f-99 9d 24 a6 f3 e8 9f cc -|
d2 44 9f 08 8b 0a 24 89-f7 c9 07 0d 25 07 ed 3e -|
75 21 19 65 dc 98 41 9d-05 12 18 88 86 16 43 49 -|
29 f2 e8 26 16 1e ce cd-32 e7 36 74 51 27 fd a2 -|
a9 62 57 60 28 e4 64 02-06 6b ff 01 ab c5 1c 25 -|
98 07 e1 40 ad 19 b7 68-66 12 4e 80 bc 83 d2 de -|
cb 7e c2 32 c7 b8 4d d6-7d dd 63 a9 95 45 c1 90 -|
c7 99 3c 0a 24 62 fc 24-15 db d3 d2 9b 5d 78 04 -|
78 d5 40 1d e3 4e e8 30-9f 56 91 71 00 86 2c 6a -|
b2 78 ec 70 d9 71 e6 aa-b1 ad 18 f9 a6 84 b7 4b -|
5f 32 b8 e3 c7 84 ef 37-fe ae 99 b5 f2 34 84 82 -|
4a b3 d0 7c 5e 25 71 89-8b 7d 6f 5f 96 7a 1d 84 -|
96 56 34 30 ce 09 d5 00-a8 ac 15 72 21 c4 71 57 -|
e5 2a 3d df 82 b8 b8 63-dc 3f 2e 99 6c c3 e3 fd -|
92 e0 26 e1 27 b8 04 71-b0 a8 d1 df 7e 24 23 b9 -|
82 01 77 dc 8f 77 54 e6-93 c8 6c 66 87 b7 aa 9d -|
66 d4 c6 2f 5e 9e e1 cf-db b2 74 0e ea a5 e0 f7 -|
00 f1 76 f7 45 2c f8 a9-3b d9 81 59 52 0f fe d9 -|
28 02 59 82 39 51 6e b9-ac f9 6a 48 73 6f 2c 4d -|
7b c0 bf be 69 ae 0e dc-8b e6 d8 9f 66 30 1e 45 -|
1d 85 23 eb a8 02 b5 ba-c2 fd a1 ff c5 55 2b a0 -|
f7 5b 24 ee 81 d8 e1 b8-02 06 85 6e 41 5a b8 07 -|
ff 65 db b4 59 89 71 95-d5 0c 2a 67 4d 57 fd 4a -|
e8 07 02 42 20 d9 f1 c6-d5 4c 53 b0 32 68 c0 dc -|
d7 5d 8f ec 24 29 00 4f-46 8d d2 99 b2 f4 06 99 -|
9a a6 31 f1 49 16 fe 94-bb 8e 15 55 06 93 16 a3 -|
2d 10 b7 b1 cf 61 78 af-93 66 5a 75 5e 97 c0 97 -|
4c ba a9 50 ac 1b d6 92-2a ac 0a 21 12 9e 4a f0 -|
40 39 4b e5 78 88 86 17-b9 eb a0 33 8a 9a fc 7c -|
91 16 d7 52 ec 05 7e 4e-90 78 5e 45 4a dd f6 f4 -|
2e 68 f7 8e fc 60 95 aa-6a 07 9c ea ce c1 d9 55 -|
3a 78 54 9a 2a 5f 47 87-18 4a 8c 6c 34 f5 b8 e2 -|
84 36 ef 0d 2e 9d 42 d9-ff 56 e2 87 0b 2f 4d 0e -|
c0 60 35 06 9f 61 9e 4e-7b 49 41 b4 fa 04 10 bd -|
f6 ad 02 d9 7c ba 06 68-bb a7 a6 8a ab ab b1 2d -|
69 2a f1 c6 7b 1b 71 b9-d0 91 82 6f a8 3c e1 a3 -|
23 3d 4e 48 74 e5 c9 c5-95 31 ad e7 a9 db 35 cd -|
02 08 2c 29 5f f9 17 86-69 8f 13 d1 ca 83 fc ac -|
55 cf 5a e6 45 af e5 bb-e7 b5 53 4e f0 63 fc 9a -|
49 f6 45 93 c6 bf d5 b3-25 e2 93 b0 a6 a7 14 80 -|
6d b2 03 15 6a ad e8 25-f1 80 d4 ba 9a 88 bc 56 -|
14 7a 4d ad c3 24 3f 4d-35 8b f6 59 5b fd c9 32 -|
1d f5 a5 53 b5 fb ba 83-29 0b 9c 62 9b 56 4b 44 -|
bc cc 19 59 7c 0b 74 d9-04 28 b3 67 ab 82 36 39 -|
55 5f 7c ed 84 c1 16 d3-9e 9c 90 9d 55 bc 3e b9 -|
63 12 f2 26 6d d7 cc 4f-01 a2 0b d9 66 60 ad ed -|
2e bd be 28 5f 4a 33 c8-e8 d4 a6 23 8a fd 66 f5 -|

```

```

28 90 81 27 a9 44 93 68-57 44 5e ba 90 12 03 15 -|
2f 69 80 55 e8 32 63 88-30 85 50 9b b6 bc bb c6 -|
fe a2 e7 32 9d 3d 7d e2-31 93 a6 4e a0 dc 11 44 -|
d2 93 32 94 1e c6 4c 28-de a2 a6 0d 14 02 74 8a -|
84 2a 03 67 35 1d 66 3e-e9 68 4a b2 92 1a 69 48 -|
bd 23 fd 70 5a fd fe 74-39 c5 fa 11 ac 04 c9 94 -|
fc 12 2e 06 04 61 8e 32-f4 f8 3b d1 d8 09 b3 e4 -|
ac 0f 3e 92 f7 75 0b 32-9b d4 8a 13 99 6a 26 77 -|
9f 34 08 a8 eb b3 3e 2a-5b 4a 44 f9 21 89 2a 09 -|
9c c7 0d 2a d8 d6 27 30-25 39 84 14 11 47 ff 60 -|
e4 7c c3 41 fd d5 34 dd-b1 1c f2 eb b1 67 04 fa -|
fd 65 5b 20 6e 28 75 a9-74 8e c4 2f f7 b2 f5 9f -|
13 44 ff b0 f0 68 b0 69-15 a6 16 a8 ac c3 06 14 -|
8e 51 99 a9 4c 19 d1 25-34 b5 79 c2 a7 bf d8 3d -|
2d 4c 33 ac 1b 6c af 10-42 41 14 02 e6 87 2b e9 -|
ec c6 b1 eb 97 d4 35 49-97 fc e2 73 f9 98 46 7c -|
f6 17 2d b5 43 07 8b 19-95 9b 65 d3 05 7e b0 68 -|
0d 6e 4b 60 ad 5c 47 6e-37 fd 3f 60 43 da b2 34 -|
00 d6 9c 6d 46 7f 41 e2-c1 1a d0 53 72 81 0b 3f -|
77 e1 bc cc 09 0f a1 1d-73 8c ac a4 48 90 80 a8 -|
50 63 6a b7 76 91 91 2f-1a 5e 83 80 e6 ae 66 77 -|
44 e0 0f 14 70 c9 d3 91-e8 d2 c4 89 a8 45 c0 3d -|
bd 09 58 e0 cd e6 5c 9e-02 94 d2 be df 94 35 f7 -|
67 96 75 88 08 59 d9 19-21 da d0 a2 74 2d 22 87 -|
37 27 6e 58 dc 8e 9f 50-d5 62 f3 4a a4 b2 fb f9 -|
3e d5 da 57 56 5c cb 0e-d1 62 4f ea 42 4a 62 b2 -|
4e 1c d1 cc 24 1b dc ac-d4 b0 2f 5d 62 87 56 3d -|
e4 03 ae 4a 7e 7d 05 fe-85 33 da 5f 36 cb 56 a1 -|
14 80 63 26 75 3a c2 1f-9c dd 8a d6 f8 a9 1a f6 -|
c2 57 e0 7b 80 55 d5 12-f1 b4 e7 1d 95 68 02 f1 -|
19 ff 74 72 32 db 6c c9-a0 2d 69 fc c0 e8 27 11 -|
b8 e0 f5 83 60 3f b4 94-e5 9d b8 fd c5 9e 50 76 -|
92 c7 33 6b e0 7e 9b 2c-3b 27 d8 a6 da b3 a1 45 -|
ef 3b 3d 76 1d 5a 43 a4-e1 82 5f 7d 0b 10 28 1c -|
b9 8f 2b cd f9 c5 93 e3-65 a5 5b 50 a9 07 16 b3 -|
45 4a 69 fc 58 12 7b 58-34 9f 6f 5e 7a c2 9f 7b -|
cf 62 62 6c 1e 7d d9 72-ca 98 1e d4 e5 c7 98 27 -|
e3 9f c5 3c 90 9e 26 ed-7e 38 2b cf c4 99 f5 b4 -|
78 48 7c 08 99 bc 80 47-b1 c5 58 60 17 84 11 f8 -|
70 a1 26 95 94 77 f4 8d-1d 47 96 df 95 1d 97 37 -|
5a 12 fb 3f cd a0 fb ac-61 62 1f ee dc 6d 2c 45 -|
5f c0 80 7d 99 62 38 f7-6d 88 d5 e2 24 9d ed a7 -|
d5 e7 1c 8a 75 9b 67 0c-64 fe e7 e2 e9 a0 94 60 -|
26 46 d8 c9 59 43 17 85-07 32 8b 0c 5a 1e 35 48 -|
46 e5 44 5b c7 07 cd 30-97 80 fa f9 eb 0d af af -|
3e be 27 43 8e 4a f2 60-eb 2b 4d 11 9a e1 d7 59 -|
78 29 26 3e 9b da 61 15-ab e8 6f 81 73 ac 9f 43 -|
c6 b1 81 73 36 4a 1e 57-2b d9 7a 06 4f b6 37 11 -|
e6 c6 90 41 a4 a6 b7 3a-7e c9 ce 50 2f 8c 07 db -|
9a 19 38 51 35 50 f3 09-33 20 62 36 a8 6c 8e f2 -|
56 8c 82 d3 fa 16 b5 3d-74 28 dc 2a e5 ae ab 77 -|
e5 22 3d 69 92 b8 56 c5-06 6b 8b da 46 18 af 93 -|
a2 0c b5 d3 d6 94 4a 29-fd 4b 48 48 a1 73 bc de -|
cb 3a 35 27 d5 40 3a 1d-5b e3 62 a5 22 a2 7b b5 -|
6a f0 5e 0a 96 5c e8 3b-41 58 a1 d5 11 2c 36 9e -|
e5 2b c9 fb 1b 37 d0 8e-cf c2 01 6f b6 21 96 9c -|
14 2f 76 19 b0 6a 9d ef-69 fd d0 03 d9 eb b7 86 -|
76 20 d3 20 6a d8 cb c8-9e b7 82 28 b2 25 a7 a2 -|
6b fd 60 b0 11 09 53 5f-79 6b 72 d7 1b 21 73 f7 -|
21 d5 5c c4 e5 52 82 73-1f 9c 95 e1 21 bf 12 67 -|
c9 41 e5 64 c4 d4 f9 a2-9b 29 df e5 a4 f3 b4 69 -|
6d 2c b0 42 e3 e6 25 a7-8f f5 12 99 8c bf bb d8 -|
a8 23 db 8e ec 37 b0 8e-4f ed 67 aa 3e f6 24 56 -|
96 e3 d4 bc 25 0b 56 -/
EncryptedLicenseInfo::pBlob

The decrypted LicenseInfo for the above data is
00 -\
00 -|

```

```

06 -|
00 -/ LicenseInfo::dwVersion = 0x060000

0e -\
00 -|
00 -|
00 -/ LicenseInfo::cbScope

6d 69 63 72 6f 73 6f 66 -\
74 2e 63 6f 6d 00 -/ LicenseInfo::pbScope

2c -\
00 -|
00 -|
00 -/ LicenseInfo::cbCompanyName

4d 00 69 00 63 00 72 00 -\
6f 00 73 00 6f 00 66 00 -|
74 00 20 00 43 00 6f 00 -|
72 00 70 00 6f 00 72 00 -|
61 00 74 00 69 00 6f 00 -|
6e 00 00 00 -/ LicenseInfo::pbCompanyName

08 -\
00 -|
00 -|
00 -/ LicenseInfo::cbProductId

41 00 30 00 32 00 00 00 ->LicenseInfo::pbProductId

99 -\
07 -|
00 -|
00 -/ LicenseInfo::cbLicenseInfo

30 82 07 95 06 09 2a 86 48 86 -\
f7 0d 01 07 02 a0 82 07-86 30 82 07 82 02 01 01 -|
31 00 30 0b 06 09 2a 86-48 86 f7 0d 01 07 01 a0 -|
82 07 6a 30 82 02 f1 30-82 01 dd a0 03 02 01 02 -|
02 08 01 9e 27 4d 68 ac-ed 20 30 09 06 05 2b 0e -|
03 02 1d 05 00 30 32 31-30 30 13 06 03 55 04 03 -|
1e 0c 00 52 00 4f 00 44-00 45 00 4e 00 54 30 19 -|
06 03 55 04 07 1e 12 00-57 00 4f 00 52 00 4b 00 -|
47 00 52 00 4f 00 55 00-50 30 1e 17 0d 37 30 30 -|
35 33 30 31 30 33 36 31-38 5a 17 0d 34 39 30 35 -|
33 30 31 30 33 36 31 38-5a 30 32 31 30 30 13 06 -|
03 55 04 03 1e 0c 00 52-00 4f 00 44 00 45 00 4e -|
00 54 30 19 06 03 55 04-07 1e 12 00 57 00 4f 00 -|
52 00 4b 00 47 00 52 00-4f 00 55 00 50 30 82 01 -|
22 30 0d 06 09 2a 86 48-86 f7 0d 01 01 01 05 00 -|
03 82 01 0f 00 30 82 01-0a 02 82 01 01 00 88 ad -|
7c 8f 8b 82 76 5a bd 8f-6f 62 18 e1 d9 aa 41 fd -|
ed 68 01 c6 34 35 b0 29-04 ca 4a 4a 1c 7e 80 14 -|
f7 8e 77 b8 25 ff 16 47-6f bd e2 34 3d 2e 02 b9 -|
53 e4 33 75 ad 73 28 80-a0 4d fc 6c c0 22 53 1b -|
2c f8 f5 01 60 19 7e 79-19 39 8d b5 ce 39 58 dd -|
55 24 3b 55 7b 43 c1 7f-14 2f b0 64 3a 54 95 2b -|
88 49 0c 61 2d ac f8 45-f5 da 88 18 5f ae 42 f8 -|
75 c7 26 6d b5 bb 39 6f-cc 55 1b 32 11 38 8d e4 -|
e9 44 84 11 36 a2 61 76-aa 4c b4 e3 55 0f e4 77 -|
8e de e3 a9 ea b7 41 94-00 58 aa c9 34 a2 98 c6 -|
01 1a 76 14 01 a8 dc 30-7c 77 5a 20 71 5a a2 3f -|
af 13 7e e8 fd 84 a2 5b-cf 25 e9 c7 8f a8 f2 8b -|
84 c7 04 5e 53 73 4e 0e-89 a3 3c e7 68 5c 24 b7 -|
80 53 3c 54 c8 c1 53 aa-71 71 3d 36 15 d6 6a 9d -|
7d de ae f9 e6 af 57 ae-b9 01 96 5d e0 4d cd ed -|
c8 d7 f3 01 03 38 10 be-7c 42 67 01 a7 23 02 03 -|
01 00 01 a3 13 30 11 30-0f 06 03 55 1d 13 04 08 -|
30 06 01 01 ff 02 01 00-30 09 06 05 2b 0e 03 02 -|

```

```

1d 05 00 03 82 01 01 00-70 db 21 2b 84 9a 7a c3 -|
b1 68 fa c0 00 8b 71 ab-43 9f b6 7b b7 1f 20 83 -|
ac 0a b5 0e ad b6 36 ef-65 17 99 86 8a 3d ba 0c -|
53 2e a3 75 a0 f3 11 3d-e7 65 4b ae 3c 42 70 11 -|
dc ca 83 c0 be 3e 97 71-84 69 d6 a8 27 33 9b 3e -|
17 3c a0 4c 64 ca 20 37-a4 11 a9 28 8f b7 18 96 -|
69 15 0d 74 04 75 2a 00-c7 a6 6a be ac b3 f2 fb -|
06 1b 6c 11 bd 96 e2 34-74 5d f5 98 8f 3a 8d 69 -|
08 6f 53 12 4e 39 80 90-ce 8b 5e 88 23 2d fd 55 -|
fd 58 3d 39 27 b3 7c 57-fe 3b ab 62 26 60 e2 d0 -|
c8 f4 02 23 16 c3 52 5d-9f 05 49 a2 71 2d 6d 5b -|
90 dd bf e5 a9 2e f1 85-8a 8a b8 a9 6b 13 cc 8d -|
4c 22 41 ad 32 1e 3b 4b-89 37 66 df 1e a5 4a 03 -|
52 1c d9 19 79 22 d4 a7-3b 47 93 a9 0c 03 6a d8 -|
5f fc c0 75 33 e5 26 da-f7 4a 77 d8 f1 30 80 39 -|
38 1e 86 1d 97 00 9c 0e-ba 00 54 8a c0 12 32 6f -|
3d c4 15 f9 50 f8 ce 95-30 82 04 71 30 82 03 5d -|
a0 03 02 01 02 02 05 03-00 00 00 0f 30 09 06 05 -|
2b 0e 03 02 1d 05 00 30-32 31 30 30 13 06 03 55 -|
04 03 1e 0c 00 52 00 4f-00 44 00 45 00 4e 00 54 -|
30 19 06 03 55 04 07 1e-12 00 57 00 4f 00 52 00 -|
4b 00 47 00 52 00 4f 00-55 00 50 30 1e 17 0d 30 -|
37 30 36 32 30 31 34 35-31 33 35 5a 17 0d 30 37 -|
30 39 31 38 31 34 35 31-33 35 5a 30 7f 31 7d 30 -|
13 06 03 55 04 03 1e 0c-00 52 00 4f 00 44 00 45 -|
00 4e 00 54 30 21 06 03-55 04 07 1e 1a 00 41 00 -|
64 00 6d 00 69 00 6e 00-69 00 73 00 74 00 72 00 -|
61 00 74 00 6f 00 72 30-43 06 03 55 04 05 1e 3c -|
00 31 00 42 00 63 00 4b-00 65 00 64 00 79 00 32 -|
00 6b 00 72 00 4f 00 34-00 2f 00 4d 00 43 00 44 -|
00 4c 00 49 00 31 00 41-00 48 00 5a 00 63 00 50 -|
00 69 00 61 00 73 00 3d-00 0d 00 0a 30 82 01 22 -|
30 0d 06 09 2a 86 48 86-f7 0d 01 01 01 05 00 03 -|
82 01 0f 00 30 82 01 0a-02 82 01 01 00 88 ad 7c -|
8f 8b 82 76 5a bd 8f 6f-62 18 e1 d9 aa 41 fd ed -|
68 01 c6 34 35 b0 29 04-ca 4a 4a 1c 7e 80 14 f7 -|
8e 77 b8 25 ff 16 47 6f-bd e2 34 3d 2e 02 b9 53 -|
e4 33 75 ad 73 28 80 a0-4d fc 6c c0 22 53 1b 2c -|
f8 f5 01 60 19 7e 79 19-39 8d b5 ce 39 58 dd 55 -|
24 3b 55 7b 43 c1 7f 14-2f b0 64 3a 54 95 2b 88 -|
49 0c 61 2d ac f8 45 f5-da 88 18 5f ae 42 f8 75 -|
c7 26 6d b5 bb 39 6f cc-55 1b 32 11 38 8d e4 e9 -|
44 84 11 36 a2 61 76 aa-4c b4 e3 55 0f e4 77 8e -|
de e3 a9 ea b7 41 94 00-58 aa c9 34 a2 98 c6 01 -|
1a 76 14 01 a8 dc 30 7c-77 5a 20 71 5a a2 3f af -|
13 7e e8 fd 84 a2 5b cf-25 e9 c7 8f a8 f2 8b 84 -|
c7 04 5e 53 73 4e 0e 89-a3 3c e7 68 5c 24 b7 80 -|
53 3c 54 c8 c1 53 aa 71-71 3d 36 15 d6 6a 9d 7d -|
de ae f9 e6 af 57 ae b9-01 96 5d e0 4d cd ed c8 -|
d7 f3 01 03 38 10 be 7c-42 67 01 a7 23 02 03 01 -|
00 01 a3 82 01 47 30 82-01 43 30 14 06 09 2b 06 -|
01 04 01 82 37 12 04 01-01 ff 04 04 01 00 05 00 -|
30 3c 06 09 2b 06 01 04-01 82 37 12 02 01 01 ff -|
04 2c 4d 00 69 00 63 00-72 00 6f 00 73 00 6f 00 -|
66 00 74 00 20 00 43 00-6f 00 72 00 70 00 6f 00 -|
72 00 61 00 74 00 69 00-6f 00 6e 00 00 00 30 56 -|
06 09 2b 06 01 04 01 82-37 12 05 01 01 ff 04 46 -|
00 30 00 00 01 00 00 00-ff 00 00 00 00 04 00 00 -|
1c 00 08 00 24 00 16 00-3a 00 01 00 41 00 30 00 -|
32 00 00 00 41 00 30 00-32 00 2d 00 36 00 2e 00 -|
30 00 30 00 2d 00 53 00-00 00 06 00 00 00 80 -|
64 80 00 00 00 00 30 6e-06 09 2b 06 01 04 01 82 -|
37 12 06 01 01 ff 04 5e-00 30 00 00 00 00 0e 00 -|
3e 00 52 00 4f 00 44 00-45 00 4e 00 54 00 00 00 -|
37 00 38 00 34 00 34 00-30 00 2d 00 30 00 30 00 -|
36 00 2d 00 35 00 38 00-36 00 37 00 30 00 34 00 -|
35 00 2d 00 37 00 30 00-33 00 34 00 37 00 00 00 -|
57 00 4f 00 52 00 4b 00-47 00 52 00 4f 00 55 00 -|
50 00 00 00 00 00 30 25-06 03 55 1d 23 01 01 ff -|

```

```

04 1b 30 19 a1 10 a4 0e-52 00 4f 00 44 00 45 00 -|
4e 00 54 00 00 00 82 05-03 00 00 00 0f 30 09 06 -|
05 2b 0e 03 02 1d 05 00-03 82 01 01 00 13 1b dc -|
89 d2 fc 54 0c ee 82 45-68 6a 72 c3 3e 17 73 96 -|
53 44 39 50 0e 0b 9f 95-d6 2c 6b 53 14 9c e5 55 -|
ed 65 df 2a eb 5c 64 85-70 1f bc 96 cf a3 76 b1 -|
72 3b e1 f6 ad ad ad 2a-14 af ba d0 d6 d5 6d 55 -|
ec 1e c3 4b ba 06 9c 59-78 93 64 87 4b 03 f9 ee -|
4c dd 36 5b bd d4 e5 4c-4e da 7b c1 ae 23 28 9e -|
77 6f 0f e6 94 fe 05 22-00 ab 63 5b e1 82 45 a6 -|
ec 1f 6f 2c 7b 56 de 78-25 7d 10 60 0e 53 42 4b -|
6c 7a 6b 5d c9 d5 a6 ae-c8 c8 52 29 d6 42 56 02 -|
ec f9 23 a8 8c 8d 89 c9-7c 84 07 fc 33 e1 1e ea -|
e2 8f 2b be 8f a9 d3 d1-e1 5e 0b dc b6 43 6e 33 -|
0a f4 2e 9d 0c c9 58 54-34 aa e1 d2 a2 e4 90 02 -|
23 26 a0 92 26 26 0a 83-b4 4d d9 4b ef eb 9d a9 -|
24 3f 92 8b db 04 7b 9d-64 91 a4 4b d2 6e 51 05 -|
08 c9 91 af 31 26 55 21-b1 ea ce a3 a4 0d 5e 4c -|
46 db 16 2d 98 dc 60 19-b8 1b b9 cd fb 31 00 -|
LicenseInfo::pbLicenseInfo
0x7f7: MACData (0x10 bytes)
ed-e8 bf d6 13 a0 f5 80 -\
4a e5 ff 85 16 fa cb 1f -/ MACData

```

4.7 SERVER UPGRADE LICENSE

The [Server Upgrade License](#) message is sent to the client to upgrade a license in its **license store**. The message type is UPGRADE_LICENSE (0x04) in the [licensing preamble](#). See section [2.2.2.7](#) for more information.

```

00000000 04 03 95 1b 09 00 7d 1b-40 06 79 f3 66 e3 1c ef .....}.@.y.f...
00000010 a3 06 7d ab 98 9a c8 a9-7b 55 57 2a 43 43 2a 85 ..}.....{UW*CC*.
00000020 33 f3 97 39 23 b7 ca 83-88 09 85 14 fe c0 65 d8 3..9#.....e.
00000030 1b 53 e2 50 64 51 db a3-cb 4e 2c 52 ac 0d b5 40 .S.PdQ...N,R...@
00000040 3d 9f c3 04 47 ff 94 de-09 80 00 c5 4b 11 5c 0e =...G.....K.\.
00000050 3a 4a f9 45 15 a1 96 7c-34 ec ae d8 16 48 c4 05 :J.E...|4....H..
00000060 d1 cf b8 d7 05 dc 2e fe-8e bf 19 1d bd 3d f9 89 .....=...
00000070 e7 38 61 e2 95 e9 38 9e-11 ee 44 c1 46 44 77 c4 .8a...8...D.FDw.
00000080 c7 ee 38 6e a4 c2 ca 87-47 ae 3f 7d ed cb f0 01 ..8n....G.?)....
00000090 22 f7 bf 0f cc 63 29 42-40 d1 65 3f 1d 8a 6d 60 ".....c)B@.e?.m`
000000A0 e7 09 2a 03 ff f5 a3 6f-d0 b0 58 6b 05 fb f1 07 ..*.....o...Xk....
000000B0 da 3a fa 92 4f bc 3f ff-2d 68 bc c8 be 59 95 fd ....O.?.-h...Y..
000000C0 60 ff 13 3f 9f da de 69-39 5d 46 b8 44 19 3d 45 `...?.i9]F.D.=E
000000D0 1c f5 18 99 d2 da fe 03-a5 6a 03 77 77 42 0a 15 .....j.wWB..
000000E0 e3 fc 27 9f b9 f7 f8 c3-20 7b 02 98 ed ec 77 6b ..'.....{....wk
000000F0 ed f2 65 3f 4d fc 3f 8d-31 92 df f3 b8 28 04 9f ..e?M.?.1....(..
00000100 f5 ac a3 e9 5e d0 cc d8-96 95 10 c7 19 3c a6 6e ....^.....<.n
00000110 26 54 7d 85 68 30 30 3c-f5 6f e0 ad dc be 1c ea &T}.h00<.o.....
00000120 33 53 46 3c 7f 7d c4 0c-1e d2 4e 5c 7c 6a d4 6d 3SF<..}....N\|j.m
00000130 92 9a 76 71 40 3d 61 bb-9e 69 94 2c 45 2e 7f 49 ..vq@=a..i.,E..I
00000140 3a ac 1e bc 16 1c 1f 4d-5f 23 eb 9f 4f 04 b0 28 :.....M_#..O..(
00000150 77 59 bd 52 23 e0 e7 79-75 13 5d a4 93 3f fb f9 wY.R#..yu.]...?..
00000160 e0 a9 69 7f 32 a8 7c 1b-87 a9 3f 12 21 ed 54 c5 ..i.2.|...?..!..T.
00000170 4c e8 c4 57 5c c5 6f 76-79 7f b7 70 9b 19 1f 64 L..W\..ovy..p....d
00000180 9e d3 17 cf 03 6c 78 fe-71 b9 c7 aa 3f 5f 25 00 .....lx.q...? %.
00000190 1d ff d1 74 65 f9 ef 3c-b3 ac 69 51 8b a3 b3 07 ...te..<..iQ....
000001A0 97 ca 0e b4 42 eb 15 13-e3 e2 9a 8e 3d 50 bd 28 .....B.....=P.(
000001B0 f9 98 08 bb aa 23 1a 64-0f 15 72 f8 3a d6 24 14 .....#.d.r.r.:$.
000001C0 b3 73 12 5a 50 b8 61 8b-98 e0 1e c3 5c cc e7 e7 .s.ZP.a.....\...
000001D0 0b 37 3f bd f9 13 25 0b-98 48 b0 a7 5b 20 5d d6 .7?...%.H..[ ]
000001E0 d4 87 93 38 b2 99 72 ae-68 81 c2 e4 cb ed 34 21 ...8..r.h.....4!
000001F0 b5 81 62 0e af ec aa 13-88 fc 88 64 2f b3 18 04 ..b.....d/...
00000200 44 c9 65 e7 e8 37 1c 43-42 c5 eb c3 0a 39 e3 df D.e..7.CB....9..
00000210 e3 04 5d 5a b2 3e 9c 3e-24 14 df 25 0d dd 3d 59 ..]Z.>.>$.%..=Y

```



```
95 -\
1b -/ LICENSE_PREAMBLE::wMsgSize = 0x1b95 bytes

0x04: EncryptedLicenseInfo (2 + 2 + 0x1b7d = 0x1b81 bytes)
09 -\
00 -/ EncryptedLicenseInfo::wBlobType = BB ENCRYPTED DATA BLOB

7d -\
1b -/ EncryptedLicenseInfo::wBlobLen = 0x1b7d bytes

The remaining part of this blob is the EncryptedLicenseInfo. The decrypted
    LicenseInfo blob and data fields can be seen in section 4.1.6: Protocol
    Examples, SERVER_NEW_LICENSE.
0x1b85: MACData
73 da-36 1e 92 c8 d0 78 12 c3 1c d3 68 a5 c6 00 -> MACData
```

5 Security

5.1 Security Considerations for Implementers

The Remote Desktop Protocol: Licensing Extension uses its own security layer in addition to the RDP security layer (see [\[MS-RDPBCGR\]](#) section 5). Certain fields of licensing messages (see sections [2.2.2.1](#) through [2.2.2.7](#)) are protected by encryption based on the **license encryption key**.

The license encryption key is different from the **session encryption key** used in the RDP security layer (see [\[MS-RDPBCGR\]](#) section 5.3.5). The license encryption key is always generated irrespective of whether RDP encryption is in effect. The client generates the license encryption key on receipt of the Server License Request message. The server generates the license encryption key on receipt of a [Client License Information](#) message or a [Client New License Request](#) message. Both client and server use the same license encryption key for the license protocol. For information on how the license encryption key is generated, see section [5.1.3](#).

5.1.1 X.509 Certificate

Certificates that **license servers** issue to **terminal servers** and **Remote Desktop clients** conform to the X.509 Version 3 format (see [\[RFC3280\]](#)) and are signed using the **object identifier (OID)** "1.3.14.3.2.29" (OID_OIWSEC_sha1RSASign) (for more information, see [\[MSDN-CAI\]](#)). The certificates generated by the license server include the following extension OIDs:

- "1.3.6.1.4.1.311.18.4"(OID_HYDRA_CERT_VERSION)
- "1.3.6.1.4.1.311.18.2"(OID_MANUFACTURER)
- "1.3.6.1.4.1.311.18.5"(OID_LICENSED_PRODUCT_INFO)
- "1.3.6.1.4.1.311.18.6"(OID_MS_LICENSE_SERVER_INFO)
- "1.3.6.1.4.1.311.18.7"(OID_PRODUCT_SPECIFIC_OID)

The license server certificate can be either a self-signed certificate or issued by the **clearing house**. For the X.509 certificate chain, see section [2.2.1.4.2](#).

5.1.2 Client and Server Random Values and Premaster Secrets

The client and server both generate a 32-byte random value using a cryptographically safe random number generator.

The server generates a 32-byte server random value and sends it to the client in the [Server License Request](#) message. The server also sends its public key embedded in a [server certificate](#) as part of the Server License Request message.

On receipt of a Server License Request message, the client generates a 32-byte client random value. It also generates a 48-byte random number called the premaster secret. The client encrypts the **premaster secret** (see section [5.1.2.1](#)) using the server's public key (embedded in the Server Certificate in the Server License Request message). The client then sends the client random value and the encrypted premaster secret to the server in a [Client New License Request](#) message or a [Client License Information](#) message, depending on whether the client possesses a license.



Figure 7: Client and server random values and premaster secret flows

For information on how the licensing encryption key is generated, see section [5.1.3](#).

5.1.2.1 Encrypting the Premaster Secret

The client uses RSA to encrypt the **premaster secret** with the public key of the server. The client obtains the public key from the **terminal server certificate**, which is the leaf certificate in the certificate chain that is obtained from the [Server Certificate \(SERVER CERTIFICATE\)](#) in the [Server License Request](#) message.

For encryption with RSA, see [\[MS-RDPBCGR\]](#) sections 5.3.1 and 5.3.4.

5.1.2.2 Decrypting the Premaster Secret

The server decrypts the encrypted **premaster secret** with its private key. For decryption with RSA, see [\[MS-RDPBCGR\]](#) sections 5.3.1 and 5.3.4.

5.1.3 Generating the Licensing Encryption and MAC Salt Keys

Both the client and the server use the licensing encryption key when necessary to encrypt and decrypt licensing message data. Both the client and the server use the method described in this section to generate the licensing encryption key. The key generating procedure is described as follows. Note that the "+" symbol is used in the following procedure to represent concatenation of the keys.

1. The client and server random values and the decrypted **premaster secret** are first used to generate a 384-bit master secret, as follows. Note that **SHA-1 hash** is used.

```

SaltedHash(S, I) = MD5(S + SHA-1 (I + S + ClientRandom + ServerRandom))
PreMasterHash(I) = SaltedHash(PremasterSecret, I)
MasterSecret = PreMasterHash('A') + PreMasterHash('BB')
               + PreMasterHash('CCC')
  
```

2. A 384-bit SessionKeyBlob is generated.

```

SaltedHash2(S, I) = MD5(S + SHA-1 (I + S + ServerRandom + ClientRandom))

MasterHash(I) = SaltedHash2(MasterSecret, I)
SessionKeyBlob = MasterHash('A') + MasterHash('BB')
                 + MasterHash('CCC')
  
```

3. The first 128 bits of the SessionKeyBlob are used to generate the MAC salt key.

```
MAC-salt-key = First128Bits(SessionKeyBlob)
```

4. The MAC salt key is used to generate the **MAC** checksum that the recipient uses to check the integrity of the licensing message.
5. The licensing encryption key is derived from the SessionKeyBlob. Note that the "+" symbol is used in the following procedure to represent concatenation of the keys.

```
FinalHash(K) = MD5(K + ClientRandom + ServerRandom)  
LicensingEncryptionKey = FinalHash(Second128Bits(SessionKeyBlob))
```

5.1.4 Encrypting Licensing Session Data

The server and the client both encrypt the licensing data with the 128-bit license encryption key (obtained as described in section [5.1.3](#)) using **RC4**. For more information, see [\[SCHNEIER\]](#).

```
EncryptedData = RC4(LicensingEncryptionKey, LicensingData)
```

5.1.5 Decrypting Licensing Session Data

The server and the client both decrypt the licensing data with the 128-bit license encryption key (obtained as specified in section [5.1.3](#)), using **RC4** (for more information, see [\[MSDN-RC4\]](#)).

```
DecryptedData = RC4(LicensingEncryptionKey, EncryptedData)
```

5.1.6 MAC Generation

Generation of a **MAC** checksum follows a method similar to that specified in [\[MS-RDPBCGR\]](#) section 5.3.6.1. Note that **SHA-1 hash** is used. Also note that the "+" symbol is used in the following procedure to represent concatenation of the keys.

```
MACData = MD5(MAC-salt-key + pad2 + SHA-1 (MAC-salt-key + pad1  
+ data-length + data-content))
```

The **Pad1** field is the value 0x36 repeated 40 times, the **pad2** field is the value 0x5C repeated 48 times, the **data-length** field is the length of the data to encrypt expressed as a little-endian (Intel-ordered) UINT32, and the **data-content** field is the full set of data to be encoded.

The MAC salt key is given by the first 128 bits of the licensing session key BLOB. For licensing session key BLOB and MAC salt key generation, see section [5.1.3](#).

5.2 Index of Security Parameters

None.

6 Appendix A: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include updates to those products.

- Windows XP operating system
- Windows Server 2003 operating system
- Windows Vista operating system
- Windows Server 2008 operating system
- Windows 7 operating system
- Windows Server 2008 R2 operating system
- Windows 8 operating system
- Windows Server 2012 operating system
- Windows 8.1 operating system
- Windows Server 2012 R2 operating system
- Windows 10 operating system
- Windows Server 2016 operating system
- Windows Server operating system

Exceptions, if any, are noted in this section. If an update version, service pack or Knowledge Base (KB) number appears with a product name, the behavior changed in that update. The new behavior also applies to subsequent updates unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms "SHOULD" or "SHOULD NOT" implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term "MAY" implies that the product does not follow the prescription.

[<1> Section 1.6](#): Client SKUs of the Windows operating system that support Remote Desktop (Windows XP Professional operating system, Windows Vista, and Windows 7) and server SKUs (Windows Server 2003, Windows Server 2008, and Windows Server 2008 R2 operating system) that run in Remote Administration mode never contact a licensing server. In such cases, the PDU flow is as specified in section [1.3.3](#).

[<2> Section 2.2.2.1](#): Windows uses 2048-bit keys. Any client/server combinations that include Windows Server 2003 and Windows XP use 512-bit keys; for example, Windows XP as a client and Windows Server 2012 as a server, or Windows 8 as a client and Windows Server 2003 as a server.

[<3> Section 2.2.2.1](#): The Windows Server 2008 terminal server does not send the [server certificate](#) if encryption is in effect and already protecting RDP traffic (for RDP security measures, see [\[MS-RDPBCGR\]](#) sections 5.3 and 5.4).

[<4> Section 2.2.2.1](#): The appropriate license refers to any valid license that can be used to connect to the terminal server in question. A terminal server requires a license of the same version or later to honor the connection. For instance, if the terminal server is running Windows Server 2008, the license store is searched for a Windows Server 2008 license or later version license.

<5> [Section 2.2.2.1.1](#): The Windows Server 2008 R2 and Windows Server 2012 terminal server minor version that is specified in the low-order field of **dwVersion** is 0, which is the same value that is used for Windows Server 2008. See [\[MSDN-OSVER\]](#) for information on getting Windows operating system version numbers.

<6> [Section 2.2.2.1.1](#): Licenses issued by Microsoft license servers contain "Microsoft Corporation" as the company name.

<7> [Section 2.2.2.1.2](#): Scope Lists originating from Microsoft terminal servers contain only one [Scope](#) structure for the entity "microsoft.com".

<8> [Section 2.2.2.2](#): Windows uses 2048-bit keys. Any client/server combinations that include Windows Server 2003 and Windows XP use 512-bit keys; for example, Windows XP as a client and Windows Server 2012 as a server, or Windows 8 as a client and Windows Server 2003 as a server.

<9> [Section 2.2.2.2](#): On Windows platforms, it is relative to the major version number of the operating system and has the following values.

Value	Meaning
CLIENT_OS_ID_WINNT_351 0x01000000	The client operating system version is 3.51.
CLIENT_OS_ID_WINNT_40 0x02000000	The client operating system version is 4.00.
CLIENT_OS_ID_WINNT_50 0x03000000	The client operating system version is 5.00.
CLIENT_OS_ID_WINNT_POST_52 0x04000000	The client operating system version is 5.20 or later.

<10> [Section 2.2.2.2](#): On Microsoft platforms, the second most significant byte can have the following values.

Value	Meaning
CLIENT_IMAGE_ID_MICROSOFT 0x00010000	The ISV for the client image is Microsoft.
CLIENT_IMAGE_ID_CITRIX 0x00020000	The ISV for the client image is Citrix.

<11> [Section 2.2.2.2](#): On Microsoft platforms, the least-significant byte of the **PlatFormId** field contains the minor operating system version.

<12> [Section 2.2.2.4](#): The **wBlobType** field is unused in **EncryptedPlatformChallenge** (part of SERVER_PLATFORM_CHALLENGE PDU), which is sent by the server to the client.

<13> [Section 2.2.2.5](#): The **wBlobType** field is unused in **EncryptedPlatformChallengeResponse** (part of CLIENT_PLATFORM_CHALLENGE_RESPONSE PDU), which is sent by the client to the server.

<14> [Section 2.2.2.5.1](#): The wClientType is WIN32_PLATFORMCHALLENGE_TYPE for all the Microsoft remote desktop clients except for those running in Windows CE where the wClientType is WINCE_PLATFORMCHALLENGE_TYPE.

<15> [Section 2.2.2.6](#): For Windows Server 2008, Windows Server 2008 R2, and Windows Server 2012, the EncryptedLicenseInfo variable (part of the [Server Upgrade License](#) PDU) sent by the server to the client has the **wBlobType** parameter set to value BB_ENCRYPTED_DATA_BLOB (0x0009). For Windows Server 2003, the value of **wBlobType** is not defined.

<16> [Section 3.1.5.3](#): In Windows XP, the RDP connection is not disconnected on receiving ST_TOTAL_ABORT as the state transition in the [license error message](#).

<17> [Section 3.1.5.3](#): All Microsoft implementations of Licensing Extension server and Licensing Extension client never send ST_RESET_PHASE_TO_START and ST_RESEND_LAST_MESSAGE.

<18> [Section 3.2.1.5](#): For example, the **ScopeList** field in a [Server License Request](#) message sent by a **terminal server** running on a Windows operating system has the following content.

```
ScopeCount = 1
ScopeCountArray[0] = "microsoft.com"
```

<19> [Section 3.3.1.3](#): Windows uses 2048-bit keys. Any client/server combinations that include Windows Server 2003 and Windows XP use 512-bit keys; for example, Windows XP as a client and Windows Server 2012 as a server, or Windows 8 as a client and Windows Server 2003 as a server.

<20> [Section 3.3.2.1](#): The Windows Vista and Windows Server 2008 Remote Desktop client implements a configurable Client Packet Wait Timer. The default duration of this timer is 300 seconds.

7 Change Tracking

This section identifies changes that were made to this document since the last release. Changes are classified as Major, Minor, or None.

The revision class **Major** means that the technical content in the document was significantly revised. Major changes affect protocol interoperability or implementation. Examples of major changes are:

- A document revision that incorporates changes to interoperability requirements.
- A document revision that captures changes to protocol functionality.

The revision class **Minor** means that the meaning of the technical content was clarified. Minor changes do not affect protocol interoperability or implementation. Examples of minor changes are updates to clarify ambiguity at the sentence, paragraph, or table level.

The revision class **None** means that no new technical changes were introduced. Minor editorial and formatting changes may have been made, but the relevant technical content is identical to the last released version.

The changes made to this document are listed in the following table. For more information, please contact dochelp@microsoft.com.

Section	Description	Revision class
6 Appendix A: Product Behavior	Added Windows Server operating system to the list of applicable products.	Major

8 Index

A

Abstract data model
 client ([section 3.1.1](#) 31, [section 3.3.1](#) 39)
 server ([section 3.1.1](#) 31, [section 3.2.1](#) 33)
[Applicability](#) 15

C

[Capability negotiation](#) 15
[CertBlob packet](#) 17
[Change tracking](#) 78
Client
 abstract data model ([section 3.1.1](#) 31, [section 3.3.1](#) 39)
 higher-layer triggered events ([section 3.1.4](#) 31, [section 3.3.4](#) 41)
 initialization ([section 3.1.3](#) 31, [section 3.3.3](#) 41)
 local events ([section 3.1.7](#) 33, [section 3.3.7](#) 43)
 message processing ([section 3.1.5](#) 31, [section 3.3.5](#) 41)
 [other local events](#) 43
 sequencing rules ([section 3.1.5](#) 31, [section 3.3.5](#) 41)
 timer events ([section 3.1.6](#) 33, [section 3.3.6](#) 43)
 timers ([section 3.1.2](#) 31, [section 3.3.2](#) 41)
[Client Licensing States](#) 40
[Client Hardware Identification packet](#) 25
[Client License Information packet](#) 24
[Client New License Request packet](#) 22
[Client Platform Challenge Response packet](#) 26

D

Data model - abstract
 client ([section 3.1.1](#) 31, [section 3.3.1](#) 39)
 server ([section 3.1.1](#) 31, [section 3.2.1](#) 33)

E

Encryption Keys ([section 3.2.1.10](#) 35, [section 3.3.1.10](#) 40)
[Examples - overview](#) 44

F

[Fields - vendor-extensible](#) 15

G

[Glossary](#) 7

H

Higher-layer triggered events
 client ([section 3.1.4](#) 31, [section 3.3.4](#) 41)
 server ([section 3.1.4](#) 31, [section 3.2.4](#) 35)

I

[Implementer - security considerations](#) 72

[Index of security parameters](#) 74

[Informative references](#) 10

Initialization

 client ([section 3.1.3](#) 31, [section 3.3.3](#) 41)
 server ([section 3.1.3](#) 31, [section 3.2.3](#) 35)

[Introduction](#) 7

L

[Licensing PDU \(TS LICENSING PDU\) message](#) 17

[Licensing PDU packet](#) 17

Local events

 client ([section 3.1.7](#) 33, [section 3.3.7](#) 43)
 server ([section 3.1.7](#) 33, [section 3.2.7](#) 39)

M

Message processing

 client ([section 3.1.5](#) 31, [section 3.3.5](#) 41)
 server ([section 3.1.5](#) 31, [section 3.2.5](#) 35)

Messages

[Licensing PDU \(TS LICENSING PDU\)](#) 17
 [syntax](#) 16
 [transport](#) 16

N

[NEW LICENSE INFO packet](#) 29

[Normative references](#) 9

O

Other local events

[client](#) 43
 [server](#) 39

[Overview \(synopsis\)](#) 10

P

[Parameters - security index](#) 74

[Platform Challenge Response Data packet](#) 27

[Preconditions](#) 15

[Prerequisites](#) 15

[Product behavior](#) 75

[PRODUCT INFO packet](#) 21

R

[References](#) 9

[informative](#) 10

[normative](#) 9

[Relationship to other protocols](#) 15

S

[Scope packet](#) 22

[SCOPE LIST packet](#) 22

Security

[implementer considerations](#) 72
 [parameter index](#) 74

Sequencing rules

- [client](#) 41
- [server](#) 35
- Server
 - abstract data model ([section 3.1.1](#) 31, [section 3.2.1](#) 33)
 - higher-layer triggered events ([section 3.1.4](#) 31, [section 3.2.4](#) 35)
 - initialization ([section 3.1.3](#) 31, [section 3.2.3](#) 35)
 - local events ([section 3.1.7](#) 33, [section 3.2.7](#) 39)
 - message processing ([section 3.1.5](#) 31, [section 3.2.5](#) 35)
 - [other local events](#) 39
 - sequencing rules ([section 3.1.5](#) 31, [section 3.2.5](#) 35)
 - timer events ([section 3.1.6](#) 33, [section 3.2.6](#) 39)
 - timers ([section 3.1.2](#) 31, [section 3.2.2](#) 35)
- [Server Licensing States](#) 35
- [Server License Request packet](#) 20
- [Server Platform Challenge packet](#) 26
- [Server Upgrade License packet](#) 28
- [Standards assignments](#) 15
- [Syntax](#) 16

T

- Timer events
 - client ([section 3.1.6](#) 33, [section 3.3.6](#) 43)
 - server ([section 3.1.6](#) 33, [section 3.2.6](#) 39)
- Timers
 - client ([section 3.1.2](#) 31, [section 3.3.2](#) 41)
 - server ([section 3.1.2](#) 31, [section 3.2.2](#) 35)
- [Tracking changes](#) 78
- [Transport](#) 16
- Triggered events - higher-layer
 - client ([section 3.1.4](#) 31, [section 3.3.4](#) 41)
 - server ([section 3.1.4](#) 31, [section 3.2.4](#) 35)

V

- [Vendor-extensible fields](#) 15
- [Versioning](#) 15

X

- [X509 Certificate Chain packet](#) 16