Network Working Group
Request for Comments: 4254
Category: Standards Track

T. Ylonen
SSH Communications Security Corp
C. Lonvick, Ed.
Cisco Systems, Inc.
January 2006

Протокол соединений SSH

The Secure Shell (SSH) Connection Protocol

Статус документа

В этом документе содержится спецификация протокола, предложенного сообществу Internet. Документ служит приглашением к дискуссии в целях развития и совершенствования протокола. Текущее состояние стандартизации протокола вы можете узнать из документа Internet Official Protocol Standards (STD 1). Документ может распространяться без ограничений.

Авторские права

Аннотация

Протокол SSH[1] используется для организации безопасного входа в удалённую систему (login) и организации иных безопасных служб через сети, не обеспечивающие защиты.

Этот документ описывает протокол соединений SSH, обеспечивающий интерактивные сеансы работы в системе, удалённое выполнение команд, перенаправление соединений TCP/IP и X11. Все эти каналы мультиплексируются в один зашифрованный туннель.

Протокол соединений SSH разработан для использования на базе транспортного уровня SSH и протоколов аутентификации пользователей.

Оглавление

1. Введение

Протокол соединений SSH разработан для использования поверх протоколов транспортного уровня и аутентификации пользователей SSH ([SSH-TRANS] и [SSH-USERAUTH]). Он обеспечивает интерактивные сеансы работы в системе, удалённое выполнение команд, а также перенаправление соединений TCP/IP и X11.

[1]Secure Shell - защищенная командная оболочка.

Имя службы (service name) для данного протокола - ssh-connection.

Этот документ следует читать только после прочтения документа по архитектуре SSH [SSH-ARCH]. В документе используется терминология и нотация из описания архитектуры без ссылок и дополнительных разъяснений.

2. Разработчики

Основными разработчиками этого комплекта документов являются: Tatu Ylonen, Tero Kivinen, Timo J. Rinne, Sami Lehtinen (все из SSH Communications Security Corp.) и Markku-Juhani O. Saarinen (университет Jyvaskyla). Darren Moffat был редактором этого комплекта документов и внес важный вклад в работу.

За годы подготовки этого документа множество людей внесло свой вклад. В их число входят: Mats Andersson, Ben Harris, Bill Sommerfeld, Brent McClure, Niels Moller, Damien Miller, Derek Fawcus, Frank Cusack, Heikki Nousiainen, Jakob Schlyter, Jeff Van Dyke, Jeffrey Altman, Jeffrey Hutzelman, Jon Bright, Joseph Galbraith, Ken Hornstein, Markus Friedl, Martin Forssen, Nicolas Williams, Niels Provos, Perry Metzger, Peter Gutmann, Simon Josefsson, Simon Tatham, Wei Dai, Denis Bider, der Mouse и Tadayoshi Kohno. Указанные в списке люди могли не участвовать в написании данного документа, но они внесли свой вклад в его подготовку.

3. Используемые в документе соглашения

Во всех документах, связанных с протоколом SSH, следует использовать ключевые слова: **необходимо** (MUST), **недопустимо** (MUST NOT), **требуется** (REQUIRED), **нужно** (SHALL), **не нужно** (SHALL NOT), **следует** (SHOULD), **не следует** (SHOULD NOT), **рекомендуется** (RECOMMENDED), **возможно** (MAY), **необязательно** (OPTIONAL) для описания уровня требования. Интерпретация этих слов описана в [RFC2119].

Ключевые слова **приватное использование** (PRIVATE USE), **иерархическое выделение** (HIERARCHICAL ALLOCATION), **выделение в соответствии с порядком запросов** (FIRST COME FIRST SERVED), **экспертное рассмотрение** (EXPERT REVIEW), **требуется спецификация** (SPECIFICATION REQUIRED), **одобрение IESG** (IESG APPROVAL), **согласование с IETF** (IETF CONSENSUS), **стандартизация** (STANDARDS ACTION) в данном документе при их использовании в контексте распределения пространства имен интерпретируются в соответствии с [RFC2434].

В данном наборе документов определяются поля протокола и возможные значения этих полей. Поля будут определяться вместе с протокольными сообщениями. Например, поле SSH_MSG_CHANNEL_DATA определяется следующим образом:

```
byte      SSH_MSG_CHANNEL_DATA
uint32    канал получателя
string    данные
```

В данном документе поля протокола будут указываться в одинарных кавычках, а значения полей – в двойных. В приведенном выше примере поле 'data' может содержать значения "foo" и "bar".

4. Глобальные запросы

Существует несколько типов запросов, оказывающих глобальное влияние на состояние удаленной стороны, независимо от каких-либо каналов. Примером такого запроса может служить запрос на перенаправление TCP/IP для заданного порта. Отметим, что клиент и сервер **могут** одновременно передавать глобальные запросы и получатель запроса **должен** реагировать подобающим образом. Формат глобальных запросов показан на рисунке.

```
byte      SSH_MSG_GLOBAL_REQUEST
string    request name in US-ASCII only
boolean   want reply
....      request-specific data follows
```

Значение request name следует соглашению по расширяемому именованию DNS, приведенному в [SSH-ARCH].

Получатель будет отвечать на этот запрос сообщением SSH_MSG_REQUEST_SUCCESS или SSH_MSG_REQUEST_FAILURE, если want reply имеет значение TRUE.

```
byte      SSH_MSG_REQUEST_SUCCESS
....      данные, специфичные для отклика
```

Данные, специфичные для отклика, обычно отсутствуют.

Если получатель не может распознать запрос или не поддерживает его, он просто передает сообщение SSH_MSG_REQUEST_FAILURE.

```
byte      SSH_MSG_REQUEST_FAILURE
```

В общем случае отклики не включают идентификаторов запроса. Инициатор запроса для обеспечения возможности связать полученные отклики с соответствующими запросами вводится **требование** о том, что отклики на запросы SSH_MSG_GLOBAL_REQUEST **должны** передаваться в том же порядке, в котором были получены сообщения с запросами. Для канальных запросов отклики, связанные с одним каналом, также **должны** передаваться с сохранением порядка. Однако отклики на канальные запросы для различных каналов **могут** передаваться с нарушением порядка.

5. Канальный механизм

Все терминальные сессии, перенаправления и т. п. являются каналами. Канал может открыть любая сторона. Множество каналов мультиплексируется в одно соединение.

Каналы идентифицируются номерами на каждой стороне, которые не обязательно совпадают для обеих сторон одного канала. Запросы на создание канала включают номер канала, заданный инициатором. Все остальные канальные запросы включают номер канала на стороне получателя.

Каналы управляются по потоку данных. Никакие данные не могут передаваться в канал, пока не будет получено сообщение о доступности пространства в окне.

5.1. Создание канала

Когда любая из сторон хочет создать новый канал, она выделяет для этого канала локальный номер. После этого другой стороне передается сообщение, содержащее выделенный номер канала и начальный размер окна.

```
byte      SSH_MSG_CHANNEL_OPEN
string    тип канала в кодировке US-ASCII
uint32    канал отправителя
uint32    начальный размер окна
uint32    максимальный размер пакета
....      данные, зависящие от типа канала
```

Тип канала (channel type) представляет собой имя (как описано в [SSH-ARCH] и [SSH-NUMBERS]) с возможным использованием расширений. Поле 'канал отправителя' содержит локальный идентификатор, заданный отправителем данного сообщения. Поле 'начальный размер окна' задает число байтов данных канала, которые могут быть переданы отправителем без подстройки размера окна. Поле 'максимальный размер пакета' указывает размер наибольшего пакета данных, который может быть передан отправителем. Например, для интерактивных соединений могут использоваться мелкие пакеты для ускорения откликов при использовании медленных соединений.

Удаленная сторона принимает решение о создании канала и возвращает сообщение о согласии (SSH_MSG_CHANNEL_OPEN_CONFIRMATION) или отказе (SSH_MSG_CHANNEL_OPEN_FAILURE).

```
byte      SSH_MSG_CHANNEL_OPEN_CONFIRMATION
uint32    канал получателя
uint32    канал отправителя
uint32    начальный размер окна
uint32    максимальный размер пакета
....      данные, зависящие от типа канала
```

Поле 'канал получателя' содержит номер канала, выделенный инициатором запроса, а поле 'канал отправителя' - номер канала, выделенный другой стороной.

```
byte      SSH_MSG_CHANNEL_OPEN_FAILURE
uint32    канал получателя
uint32    код причины
string    описание в кодировке ISO-10646 UTF-8 [RFC3629]
string    тег языка [RFC3066]
```

Если получатель сообщения SSH_MSG_CHANNEL_OPEN не поддерживает заданный тип канала, он просто возвращает отклик SSH_MSG_CHANNEL_OPEN_FAILURE. Клиент может включить строку описания причины отказа для пользователя. В этом случае клиентские программы должны принимать меры предосторожности, описанные в [SSH-ARCH].

Имя	Код причины
SSH_OPEN_ADMINISTRATIVELY_PROHIBITED	1
SSH_OPEN_CONNECT_FAILED	2
SSH_OPEN_UNKNOWN_CHANNEL_TYPE	3
SSH_OPEN_RESOURCE_SHORTAGE	4

Значения поля 'код причины' в сообщениях SSH_MSG_CHANNEL_OPEN_FAILURE показаны в таблице справа. Отметим, что коды для удобства приведены в десятичном формате, но на практике используются значения типа uint32.

Запросы на выделение новых значений кодов причин SSH_MSG_CHANNEL_OPEN ('reason code') из диапазона 0x00000005 - 0xFDFFFFFF и тектов связанных с ними описаний ('description') **должны** удовлетворяться по согласованию с IETF, как описано в [RFC2434]. Агентство IANA не будет выделять значений кодов причин отказа для канальных соединений (Channel Connection Failure 'reason code') из диапазона of 0xFE000000 - 0xFFFFFFFF. Эти значения выделены для **приватного использования**, как описано в [RFC2434].

Несмотря на то, что IANA не контролирует значения кодов из диапазона 0xFE000000 - 0xFFFFFFFF, этот диапазон разделен на две части, администрируемые, как описано ниже.

- Диапазон 0xFE000000 - 0xFEFFFFFF используется совместно с локально выделенными каналами. Например, если предлагается канал типа 'channel type' = "example_session@example.com" и организация его завершается отказом, отклик будет содержать код причины, выделенный IANA (как сказано выше, из диапазона 0x00000001 - to xFDFFFFFF), или выделенное локально значение из диапазона 0xFE000000 - 0xFEFFFFFF. Естественно, что если сервер не понимает 'channel type', даже определенного локально типа, в качестве кода причины (если он передается) **должно** использоваться значение 0x00000003, как описано выше. Если сервер понимает тип канала, но создать канал не удается, серверу **следует** отвечать с выделенным локально кодом причины, соответствующим предложенному локальному типу 'channel type'. Предполагается, что на практике сначала будет предприниматься попытка использовать выделенные IANA значения 'reason code', а при отсутствии нужных выделять свои значения кодов причины.
- Для значений, начинающихся с 0xFF нет ограничений или предложений по использованию. При использовании таких значений интероперабельность не предполагается. Естественно, что этот поддиапазон предназначен для экспериментов.

5.2. Передача данных

```
byte      SSH_MSG_CHANNEL_WINDOW_ADJUST
uint32    канал получателя
uint32    число добавляемых байтов
```

Размер окна задает число байтов, которые другая сторона может передать до того, как она должна будет дождаться подстройки размера окна. Обе стороны используют для подстройки размера окна сообщение, показанное на рисунке.

После приема такого сообщения получатель **может** передать на указанное число байтов больше, чем было дозволено раньше. Реализации **должны** корректно работать с окнами размером до 2^{32}-1 байтов. Увеличивать размер окна сверх этого значения **недопустимо**.

```
byte      SSH_MSG_CHANNEL_DATA
uint32    канал получателя
string    данные
```

Передача данных осуществляется с помощью сообщений, показанных на рисунке.

Максимальное число данных в сообщении определяется меньшим из значений максимального размера пакетов для канала и текущего размера окна. После передачи сообщения текущий размер окна уменьшается на объем переданной информации. Обе стороны **могут** игнорировать любые данные, которые были переданы после опустошения дозволенного окна.

Предполагается, что реализации будут вносить те или иные ограничесния на размер пакетов транспортного уровня SSH (предельный размер принимаемых пакетов **должен** быть не менее 32768 байтов, как описано в [SSH-TRANS]). Для реализаций уровня соединений SSH

- **недопустимо** анонсировать максимальный размер пакетов, который будет приводить к увеличению размера пакетов транспортного уровня сверх значения, которое этот уровень будет принимать;
- **недопустимо** генерировать пакеты данных, размер которых превышает размер пакетов, которые будет передавать транспортный уровень, даже в тех случаях, когда удаленная сторона может принимать пакеты такого размера.

```
byte      SSH_MSG_CHANNEL_EXTENDED_DATA
uint32    канал получателя
uint32         data_type_code  (код  типа
данных)
string    данные
```

Кроме того, некоторые каналы могут передавать разные типы данных (примером могут служить данные стандартного вывода ошибок (stderr) в интерактивных сессиях). Такие данные можно передавать в сообщениях SSH_MSG_CHANNEL_EXTENDED_DATA, где тип данных указывается отдельным целочисленным идентификатором (см. врезку справа). Доступные типы и их интерпретация зависят от типа канала.

Данные, передаваемые в таких сообщениях потребляют пространство окна, как и обычные данные.

Имя	data_type_code
SSH_EXTENDED_DATA_STDERR	1

В настоящее время определен только один тип данных (см. таблицу справа). Отметим, что значение 'data_type_code' приведено в десятичном формате, хотя на практике используется тип uint32.

Значения типа для расширенных каналов данных (Extended Channel Data Transfer 'data_type_code') **должны** выделяться последовательно. Запросы на выделение новых значений 'data_type_code' из диапазона 0x00000002 — 0xFDFFFFFF и связанных с ними строк 'data' **должны** удовлетворяться по процедуре IETF CONSENSUS, как описано в [RFC2434]. Агентство IANA не будет выделять значений 'data_type_code' из диапазона 0xFE000000 - 0xFFFFFFFF. Эти значения 'data_type_code' выделены **для приватного использования**, как описано в [RFC2434]. Как было отмечено выше, актуальные инструкции для IANA содержатся в [SSH-NUMBERS].

5.3. Закрытие канала

```
byte      SSH_MSG_CHANNEL_EOF
uint32    канал получателя
```

Когда одна из сторон уже не будет передавать данные в канал, ей **следует** отправить на другою сторону сообщение SSH_MSG_CHANNEL_EOF.

На такие сообщения явных откликов не передается. Однако приложение может передать сигнал EOF[1] на другую сторону канала. Отметим, что канал остается открытым после передачи этого сообщения и данные в обратном направлении могут по-прежнему передаваться. Сообщение не расходует пространства окна и может передаваться даже при пустом окне.

```
byte      SSH_MSG_CHANNEL_CLOSE
uint32    канал получателя
```

Когда любая из сторон желает разорвать канал, она передает сообщение SSH_MSG_CHANNEL_CLOSE. Приняв такое сообщение, получатель **должен** отправить в ответ сообщение SSH_MSG_CHANNEL_CLOSE, если оно уже не было передано. Канал считается закрытым для данной стороны, когда она передала и приняла сообщение SSH_MSG_CHANNEL_CLOSE; после этого освободившийся номер канала можно использовать снова. Любая из сторон **может** передать сообщение SSH_MSG_CHANNEL_CLOSE без предварительной отправки или приема сообщения SSH_MSG_CHANNEL_EOF.

Сообщение не расходует пространства окна и может передаваться даже при пустом окне.

Рекомендуется все данные, отправленные до этого сообщения доставить конечному получателю, если это возможно.

5.4. Специфичные для канала запросы

Многие типы каналов имеют расширения, характерные для данного значения 'channel type'. Примером может служить запрос pty (псевдотерминал) для интерактивной сессии.

Все специфические для каналов запросы имеют одинаковый формат (см. рисунок).

[1] End of File - конец файла. *Прим. перев.*

```
byte      SSH_MSG_CHANNEL_REQUEST
uint32    канал получателя
string      тип запроса в кодировке US-
ASCII
boolean   нужен ответ
....      зависящие от типа данные
```

Если поле 'want reply' (нужен ответ) имеет значение FALSE, отклик в ответ на запрос не будет передаваться. В противном случае получатель запроса будет передавать SSH_MSG_CHANNEL_SUCCESS, SSH_MSG_CHANNEL_FAILURE или специфичное для канала сообщение. Если запрос не распознан или не поддерживается, возвращается сообщение SSH_MSG_CHANNEL_FAILURE.

Сообщение не расходует пространства окна и может передаваться даже при пустом окне. Значения типа запроса ('request type') являются локальными для каждого типа каналов.

Клиент может передавать последующие сообщения, не ожидая отклика на такой запрос.

Имена 'request type' следуют соглашению о расширенном именовании DNS, описанному в [SSH-ARCH] и [SSH-NUMBERS].

```
byte      SSH_MSG_CHANNEL_SUCCESS
uint32    канал получателя

byte      SSH_MSG_CHANNEL_FAILURE
uint32    канал получателя
```

Эти сообщения не расходуют пространства окна и могут передаваться даже при пустом окне.

6. Интерактивные сеансы

Сессия представляет собой удаленное выполнение программы. Программой может быть командный процессор (shell), приложение, системная команда или некая встроенная подсистема. Программа может иметь терминал tty и может использовать перенаправление X11. Одновременно могут быть активны множество сессий.

6.1. Открытие сессии

Сессия начинается с передачи сообщения (см. рисунок).

```
byte      SSH_MSG_CHANNEL_OPEN
string    "session"
uint32    канал отправителя
uint32    начальный размер окна
uint32    максимальный размер пакета
```

Реализациям клиентов **следует** отвергать любые запросы на открытие сеансовых каналов для осложнения организации атак на них со стороны поврежденных серверов.

6.2. Запрос псевдотерминала

Для запроса псевдотерминала в сессии используется сообщение, показанное на рисунке.

```
byte      SSH_MSG_CHANNEL_REQUEST
uint32    recipient channel
string    "pty-req"
boolean   want_reply
string    значение переменной среды TERM (например, vt100)
uint32    ширина терминала в символах (например, 80)
uint32    высота терминала в строках (например, 24)
uint32    ширина терминала в пикселях (например, 640)
uint32    высота терминала в пикселях (например, 480)
string    кодированные режимы терминала (encoded terminal modes)
```

Кодированные режимы терминала ('encoded terminal modes') описаны в разделе 8. Параметры нулевого размера **должны** игнорироваться. Размеры терминала в символах и строках имеют более высокий приоритет, нежели размеры в пикселях. Размеры в пикселях указывают доступную для вывода область терминального окна.

Параметры размеров передаются только для информации.

Клиентам **следует** игнорировать запросы pty.

6.3. Перенаправление X11

6.3.1. Запрос перенаправления X11

```
byte      SSH_MSG_CHANNEL_REQUEST
uint32    канал получателя
string    "x11-req"
boolean   нужен ответ
boolean   одно соединение
string    протокол аутентификации x11
string    cookie для аутентификации x11
uint32    номер экрана x11
```

Перенаправление X11 для сессии можно запросить, передав сообщение SSH_MSG_CHANNEL_REQUEST.

В качестве пол 'x11 authentication cookie' р**екомендуется** передавать случайное поддельное значение cookie, которое будет проверяться и заменяться реальным значением при получении запроса на соединение.

Перенаправление X11 следует останавливать при закрытии сеансового канала. Однако уже организованные перенаправления не будут автоматически закрываться при разрыве сеансового канала.

Если поле 'single connection' (одно соединение) имеет значение TRUE, перенаправлять следует только одно соединение. После его направления или после разрыва сеансового канала другие соединения перенаправляться не будут.

Поле 'x11 authentication protocol' (протокол аутентификации) задает имя используемого X11 метода аутентификации (например, "MIT-MAGIC-COOKIE-1").

Значение поля 'x11 authentication cookie' **должно** указываться в шестнадцатеричном формате.

X Protocol документирован в [SCHEIFLER].

6.3.2. Каналы X11

```
byte      SSH_MSG_CHANNEL_OPEN
string    "x11"
uint32    канал отправителя
uint32    начальный размер окна
uint32    максимальный размер пакета
string    адрес инициатора (например, "192.168.7.38")
uint32    порт инициатора
```

Каналы X11 создаются с помощью запросов на организацию канала (см. врезку справа). Созданные каналы не зависят от сессии и закрытие сеансового канала не закрывает каналы перенаправления X11.

Получателю запроса следует передать в ответ SSH_MSG_CHANNEL_OPEN_CONFIRMATION или SSH_MSG_CHANNEL_OPEN_FAILURE.

Реализации **должны** отвергать любые запросы на организацию канала X11, если в них не запрашивается перенаправление X11.

6.4. Передача переменных окружения

```
byte      SSH_MSG_CHANNEL_REQUEST
uint32    канал получателя
string    "env"
boolean   нужен ответ
string    имя переменной
string    значение переменной
```

Переменные окружения могут передаваться команде/командному процессору, который будет запущен позже. Неконтролируемая установка переменных окружения в привилегированных процессах может приводить к рискам безопасности. Реализациям рекомендуется поддерживать список разрешенных имен переменных или устанавливать переменные окружения лишь после того, как серверный процесс откажется от значимых привилегий.

6.5. Запуск командного процессора или команды

После организации сеанса на удаленной стороне запускается программа, в качестве которой может служить командный процессор, прикладная программа или подсистема с независимым от хоста именем. Для каждого канала может быть выполнен только один из показанных ниже запросов.

```
byte      SSH_MSG_CHANNEL_REQUEST
uint32    канал получателя
string    "shell"
boolean   нужен ответ
```

Это сообщение запрашивает установленную по умолчанию (обычно задается в файле /etc/passwd для UNIX-систем) для пользователя командную оболочку (shell).

```
byte      SSH_MSG_CHANNEL_REQUEST
uint32    канал получателя
string    "exec"
boolean   нужен ответ
string    команда
```

Данное сообщение будет запрашивать у сервера запуск на выполнение заданной команды. Поле команды ('command') может содержать полный путь. **Должны** приниматься обычные меры предосторожности против несанкционированного выполнения команд.

```
byte      SSH_MSG_CHANNEL_REQUEST
uint32    канал получателя
string    "subsystem"
boolean   нужен ответ
string    имя подсистемы
```

Последний вариант является запросов на запуск предопределенной подсистемы. Предполагается, что это будет включать обычный перенос файлов и, возможно, другие функции. Разработчики могут также разрешить включение дополнительных механизмов. Поскольку для запуска подсистемы обычно используется пользовательский командный процессор (shell) протоколу подсистемы рекомендуется использовать "magic cookie" в начале протокольной транзакции, чтобы отличить его от произвольного вывода, создаваемого сценариями инициализации оболочки и т. п. Побочный вывод командного процессора может быть отфильтрован сервером или клиентом.

Серверу **не следует** прерывать работу стека протоколов при запуске командного процессора или программы. Весь ввод и вывод **следует** перенаправлять в канал или шифрованный туннель.

Рекомендуется запрашивать и проверять отклики на описанные выше запросы. Клиентам **следует** игнорировать такие запросы.

Имена подсистем должны следовать соглашению о расширенном именовании DNS приведенному в [SSH-NUMBERS].

6.6. Передача данных в сессии

Передача данный в сессии осуществляется с помощью пакетов SSH_MSG_CHANNEL_DATA и SSH_MSG_CHANNEL_EXTENDED_DATA с использованием механизма окна. Для стандартного вывода ошибок (stderr) определен расширенный тип данных SSH_EXTENDED_DATA_STDERR.

6.7. Сообщение об изменении размеров терминального окна

При изменении размеров терминального окна на клиентской стороне клиент **может** передать другой стороне сообщение с информацией о новых размерах (см. рисунок).

```
byte      SSH_MSG_CHANNEL_REQUEST
uint32    канал получателя
string    "window-change"
boolean   FALSE
uint32    ширина терминала в символах
uint32    высота терминала в строках
uint32    ширина терминала в пикселях
uint32    высота терминала в пикселях
```

Передавать отклики на такие сообщения **не следует**.

6.8. Локальное управление потоком данных

На многих системах можно определить, использует ли псевдотерминал управление потоком данных Control-S/Control-Q. При включенном управлении потоком зачастую желательно контролировать поток на клиентской стороне для ускорения откликов на запросы пользователя. Это достигается с помощью показанного здесь уведомления. Изначально за управление потоком данных отвечает сервер (напомним, что клиентом является сторона, инициировавшая сессию, а сервером - другая сторона).

```
byte      SSH_MSG_CHANNEL_REQUEST
uint32    канал получателя
string    "xon-xoff"
boolean   FALSE
boolean   клиенту разрешено
```

Показанное на врезке слева сообщение используется сервером для информирования клиента о возможности (или невозможности) управлять потоком данных (обработка Control-S/Control-Q). Если управление разрешено для клиента ('client can do' = TRUE), он может использовать Control-S и Control-Q для управления потоком данных. Клиент **может** игнорировать это сообщение.

Отклик на такое сообщение не передается.

6.9. Сигналы

```
byte      SSH_MSG_CHANNEL_REQUEST
uint32    канал получателя
string    "signal"
boolean   FALSE
string    имя сигнала (без префикса "SIG")
```

Сигналы могут передаваться процессу или службе на удаленной стороне с помощью показанного на рисунке сообщения. Некоторые системы могут не поддерживать часть сигналов - в таких случаях им **следует** игнорировать сообщение.

Значения поля «имя сигнала» ('signal name') представляются, как описано ниже для сообщений SSH_MSG_CHANNEL_REQUEST, использующих "exit-signal".

6.10. Возврат состояния при завершении

```
byte      SSH_MSG_CHANNEL_REQUEST
uint32    канал получателя
string    "exit-status"
boolean   FALSE
uint32    exit_status
```

Когда работа программы на другой стороне прерывается, может быть передано показанное на рисунке сообщение для возврата состояния по завершении команды. Возврат этого состояния **рекомендуется** выполнять. Подтверждений для сообщения не передается. После приема сообщения канал необходимо закрыть путем передачи сообщения SSH_MSG_CHANNEL_CLOSE.

Клиент **может** игнорировать сообщение с кодом завершения программы.

```
byte      SSH_MSG_CHANNEL_REQUEST
uint32    канал получателя
string    "exit-signal"
boolean   FALSE
string    имя сигнала (без префикса "SIG")
boolean   core dumped
string    сообщение об ошибке в кодировке ISO-10646 UTF-8
string    тег языка [RFC3066]
```

Работа удаленной программы может быть также прекращена «жестко» по сигналу. Для индикации таких случаев используется специальный вариант сообщения, показанный на врезке справа. Нулевое значение поля 'exit_status' обычно означает успешное завершение команды.

Поле «имя сигнала» ('signal name') может содержать одно из приведенных ниже значений (заимствовано из [POSIX]).

```
ABRT
ALRM
FPE
HUP
ILL
INT
KILL
PIPE
QUIT
SEGV
TERM
USR1
USR2
```

Дополнительные значения имени сигнала ('signal name') **могут** передаваться в формате "sig-name@xyz", где "sig-name" и "xyz" могут быть произвольными строками по усмотрению разработчика (за исключением символа @). Однако предлагается при использовании конфигурационного сценария ('configure') все обнаруженные нестандартные значения 'signal name' представлять в формате "SIG@xyz.config.guess", где "SIG" - значение 'signal name' без префикса "SIG", а "xyz" - тип хоста, как определено в "config.guess".

Поле сообщения об ошибке ('error message') содержит дополнительную текстовую информацию об ошибке. Текст может включать множество строк, разделенных символами CRLF (возврат каретки - перевод строки). Клиентская программ **может** отображать текст сообщения об ошибке пользователю. Если это осуществляется, клиентской программе следует принимать меры предосторожности, рассмотренные в [SSH-ARCH].

7. Перенаправление для портов TCP/IP

7.1. Запрос перенаправления для порта

Участнику не требуется явно запрашивать перенаправление со своей стороны на другое направление. Однако, если он желает, чтобы соединение с портом на другой стороне перенаправлялось на локальную сторону, это нужно запрашивать явно (см. сообщение на рисунке).

```
byte      SSH_MSG_GLOBAL_REQUEST
string    "tcpip-forward"
boolean   нужен ответ
string    адрес для привязки (например, "0.0.0.0")
uint32    номер порт для привязки
```

Поля адреса ('address to bind') и порта ('port number to bind') задают IP-адрес (или доменное имя) и номер порта, для которого будет восприниматься перенаправление. Некоторые строки, используемые в поле привязки адреса, имеют специальное значение.

- "" означает, что соединения будут восприниматься для всех семейств протоколов, поддерживаемых реализацией SSH;
- "0.0.0.0" означает, что прослушиваются все адреса IPv4;
- "::" означает, что прослушиваются все адреса IPv6;
- "localhost" означает, что прослушиваются все семейства протоколов, поддерживаемые реализацией SSH, но только на адресах loopback ([RFC3330] и [RFC3513]);
- "127.0.0.1" и "::1" указывает прослушивание на loopback-интерфейсах для IPv4 и IPv6, соответственно.

Отметим, что клиент может фильтровать соединения на основе информации, переданной в запросе на организацию.

Реализациям следует разрешать перенаправление для привилегированных портов только привилегированным пользователям.

Клиентским реализациям **следует** отвергать запросы на перенаправление портов, поскольку такие запросы обычно передаются только клиентами.

```
byte      SSH_MSG_REQUEST_SUCCESS
uint32    номер порта, который будет привязан на сервере
```

Если клиент предает значение 0 в качестве номера порта для привязки и поле запроса отклика ('want reply') имеет значение TRUE, сервер выделяет следующий доступный непривилегированный номер порта и отвечает сообщением, показанным на врезке справа. В остальных случаях специфичные для отклика данные отсутствуют.

```
byte      SSH_MSG_GLOBAL_REQUEST
string    "cancel-tcpip-forward"
boolean   нужен ответ
string    адрес для привязки (например, "127.0.0.1")
uint32    номер порт для привязки
```

Для отмены перенаправления порта может использоваться сообщение, показанное на врезке слева. Отметим, что запросы на организацию канала могут приниматься, пока не получен отклик на такое сообщение.

Клиентским реализациям **следует** отвергать эти, поскольку они обычно передаются только клиентами.

7.2. Каналы перенаправления TCP/IP

Когда соединение приходит в порт, для которого было запрошено удаленное перенаправление, открывается канал для перенаправления порта на другую сторону (см. рисунок).

```
byte      SSH_MSG_CHANNEL_OPEN
string    "forwarded-tcpip"
uint32    канал отправителя
uint32    начальный размер окна
uint32    максимальный размер пакета
string    адрес, который был соединен
uint32    порт, который был соединен
string    IP-адрес инициатора
uint32    порт инициатора
```

Реализации **должны** отвергать такие сообщения, если они ранее не запрашивали удаленное перенаправление для порта TCP/IP с указанным номером.

```
byte      SSH_MSG_CHANNEL_OPEN
string    "direct-tcpip"
uint32    канал отправителя
uint32    начальный размер окна
uint32    максимальный размер пакета
string    хост для соединения
uint32    порт для соединения
string    IP-адрес инициатора
uint32    порт инициатора
```

Когда соединение приходит в локально перенаправляемый порт TCP/IP, на другую сторону передается пакет, показанный на рисунке. Отметим, что такие сообщения **могут** также передаваться для портов, по отношению к которым перенаправление не запрашивалось явно. Принимающая сторона должна принять решение о возможности перенаправления.

Поля хоста ('host to connect') и порта ('port to connect') для соединения задают хост TCP/IP и номер порта, с которым получателю следует организовать канал. Хост может быть задан адресом IP или доменным именем.

Поле адреса инициатора ('originator IP address') содержит числовое представление IP-адреса машины, отправившей запрос на соединение, а поле 'originator port' указывает номер порт на этом хосте.

Перенаправляемые каналы TCP/IP независимы от каких-либо сессий и закрытие сеансового канала не оказывает влияния на закрытие перенаправляемых соединений.

Клиентским реализациям **следует** отвергать прямые запросы на открытие соединений TCP/IP из соображений безопасности.

8. Кодирование терминальных режимов

Все кодированные режимы терминала ('encoded terminal modes'), передаваемые в запросах pty, представляются в форме потока байтов. Это сделано для того, чтобы кодирование можно было передавать через различные среды. Поток состоит из пар «код-аргумент» (opcode-argument), где opcode выражается одним байтом. Коды от 1 до 159 используют один аргумент типа uint32. Коды 160 - 255 в настоящее время не определены и вызывают прекращение обработки (их следует указывать только после всех прочих данных). Байтовый поток завершается кодом TTY_OP_END (0x00).

Клиенту **следует** помещать в поток все режимы, которые ему известны, а сервер **может** игнорировать любые неизвестные ему режимы. Это дает относительную независимость от аппаратной реализации по крайней мере для систем с интерфейсом tty в стиле POSIX. Протокол может поддерживать и другие системы, но от клиента может потребоваться указание приемлемых значений множества параметров, чтобы сервер pty получил набор значений для установки подходящего режима (сервер оставляет для всех не заданных битов режима принятые по умолчанию значения, а работоспособны не все комбинации битов).

Именование кодов в основном следует флагам терминальных режимов POSIX. В таблице приведены определенные к настоящему моменту коды. Отметим, что в таблице для удобства восприятия даны десятичные значения, а на практике используются байты.

Код	Имя	Описание
0	TTY_OP_END	Показывает завершение кодов.
1	VINTR	Символ прерывания (255, если не задан). Аналогичен других символам. Не все такие символы поддерживаются в каждой системе.
2	VQUIT	Символ завершения (в системах POSIX передается сигнал SIGQUIT).
3	VERASE	Удалить символ слева от курсора.
4	VKILL	Удалить полностью текущую строку ввода.
5	VEOF	Символ завершения файла (с терминала передается EOF).
6	VEOL	Символ завершения строки в дополнение к символу возврата каретки и/или перевода строки.
7	VEOL2	Дополнительный символ завершения строки.
8	VSTART	Продолжение вывода после паузы (обычно Control-Q).
9	VSTOP	Пауза при выводе (обычно Control-S).

10	VSUSP	Приостановка текущей программы.
11	VDSUSP	Другой символ приостановки.
12	VREPRINT	Перепечатка текущей строки ввода.
13	VWERASE	Удалить слово слева от курсора.
14	VLNEXT	Ввести следующий набранный символ, даже если он относится к специальным.
15	VFLUSH	Символ очистки вывода.
16	VSWTCH	Переключение на другой уровень командной оболочки.
17	VSTATUS	Печать строки состояния системы (загрузка, команда, pid и т. п.).
18	VDISCARD	Переключение очистки терминального вывода.
30	IGNPAR	Флаг игнорирования четности. Для параметра **следует** устанавливать значение 0, если флаг имеет значение FALSE и 1 при флаге TRUE.
31	PARMRK	Маркировка ошибок четности и кадрирования.
32	INPCK	Разрешить проверку ошибок четности.
33	ISTRIP	Вырезать восьмой бит символов.
34	INLCR	Отображать на входе NL в CR.
35	IGNCR	Игнорировать CR на входе.
36	ICRNL	Отображать на входе CR в NL.
37	IUCLC	Переводить символы верхнего регистра в символы нижнего.
38	IXON	Разрешить управление потоком на выходе.
39	IXANY	Любой символ будет возобновлять работу после остановки.
40	IXOFF	Разрешить управление потоком на входе.
41	IMAXBEL	Сигнал звонка при заполнении входной очереди.
50	ISIG	Разрешить сигналы INTR, QUIT, [D]SUSP.
51	ICANON	Канонизировать строки ввода.
52	XCASE	Разрешить ввод и вывод символов верхнего регистра путем добавления перед символом нижнего регистра знака \.
53	ECHO	Разрешить эхо-вывод.
54	ECHOE	Визуально уничтожать символы.
55	ECHOK	Символ отбрасывания текущей строки.
56	ECHONL	Вывести NL, если эхо-вывод отключен.
57	NOFLSH	Не очищать после прерывания.
58	TOSTOP	Остановить вывод из фоновых заданий.
59	IEXTEN	Разрешить расширения.
60	ECHOCTL	Отображение управляющих символов, как ^(Char).
61	ECHOKE	Визуальное удаление уничтоженной строки.
62	PENDIN	Повторный набор ожидающего ввода.
70	OPOST	Разрешить обработку вывода.
71	OLCUC	Преобразовать символы нижнего регистра в верхний.
72	ONLCR	Отображать на входе NL в CR-NL.
73	OCRNL	Преобразовать символ возврата каретки в новую строку (выход).
74	ONOCR	Преобразовать символ новой строки в «возврат каретки - новая строка» (выход).
75	ONLRET	Символ новой строки выполняет возврат каретки (выход).
90	CS7	7-битовый режим.
91	CS8	8-битовый режим.
92	PARENB	Четность включена.
93	PARODD	Нечетность.
128	TTY_OP_ISPEED	Задает входную скорость в битах в секунду.
129	TTY_OP_OSPEED	Задает выходную скорость в битах в секунду.

9. Номера сообщений

В таблице приведен список сообщений с номерами.

Сообщение	Номер
SSH_MSG_GLOBAL_REQUEST	80
SSH_MSG_REQUEST_SUCCESS	81
SSH_MSG_REQUEST_FAILURE	82
SSH_MSG_CHANNEL_OPEN	90
SSH_MSG_CHANNEL_OPEN_CONFIRMATION	91
SSH_MSG_CHANNEL_OPEN_FAILURE	92
SSH_MSG_CHANNEL_WINDOW_ADJUST	93
SSH_MSG_CHANNEL_DATA	94
SSH_MSG_CHANNEL_EXTENDED_DATA	95
SSH_MSG_CHANNEL_EOF	96
SSH_MSG_CHANNEL_CLOSE	97
SSH_MSG_CHANNEL_REQUEST	98
SSH_MSG_CHANNEL_SUCCESS	99
SSH_MSG_CHANNEL_FAILURE	100

10. Взаимодействие с IANA

Этот документ является частью набора связанных документов. Вопросы согласования с агентством IANA для протокола SSH, связанные с [SSH-ARCH], [SSH-TRANS], [SSH-USERAUTH] и данным документом, детализированы в [SSH-NUMBERS].

11. Вопросы безопасности

Предполагается, что этот протокол будет работать на основе защищенного транспорта с аутентификацией. Предполагается также, что аутентификация пользователей и защита от атак на сетевом уровне обеспечивается нижележащими протоколами.

Полное рассмотрение вопросов безопасности приведено в документе [SSH-ARCH]. Специфичной для данного документа является **рекомендация** запрещать в реализациях все потенциально опасные функции (например, перенаправления агентов, X11 и TCP/IP) если ключ хоста изменился без уведомления или объяснения.

12. Литература

12.1. Нормативные документы

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

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

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

[SSH-NUMBERS] Lehtinen, S. and C. Lonvick, Ed., "The Secure Shell (SSH) Protocol Assigned Numbers", RFC 4250, January 2006.

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

[RFC2434] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 2434, October 1998.

[RFC3066] Alvestrand, H., "Tags for the Identification of Languages", BCP 47, RFC 3066, January 2001.

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

12.2. Дополнительная литература

[RFC3330] IANA, "Special-Use IPv4 Addresses", RFC 3330, September 2002.

[RFC3513] Hinden, R. and S. Deering, "Internet Protocol Version 6 (IPv6) Addressing Architecture", RFC 3513, April 2003.

[SCHEIFLER] Scheifler, R., "X Window System : The Complete Reference to Xlib, X Protocol, Icccm, Xlfd, 3rd edition.", Digital Press ISBN 1555580882, February 1992.

[POSIX] ISO/IEC, 9945-1., "Information technology – Portable Operating System Interface (POSIX)-Part 1: System Application Program Interface (API) C Language", ANSI/IEE Std 1003.1, July 1996.

Адреса авторов

Tatu Ylonen

SSH Communications Security Corp

Valimotie 17

00380 Helsinki

Finland

EMail: ylo@ssh.com

Chris Lonvick (редактор)

Cisco Systems, Inc.

12515 Research Blvd.

Austin 78759

USA

EMail: clonvick@cisco.com

Перевод на русский язык

Николай Малых

nmalykh@protokols.ru

Торговые марки

ssh – торговый знак, зарегистрированный в США и/или других странах.

Подтверждение

Финансирование функций RFC Editor обеспечено IETF Administrative Support Activity (IASA).