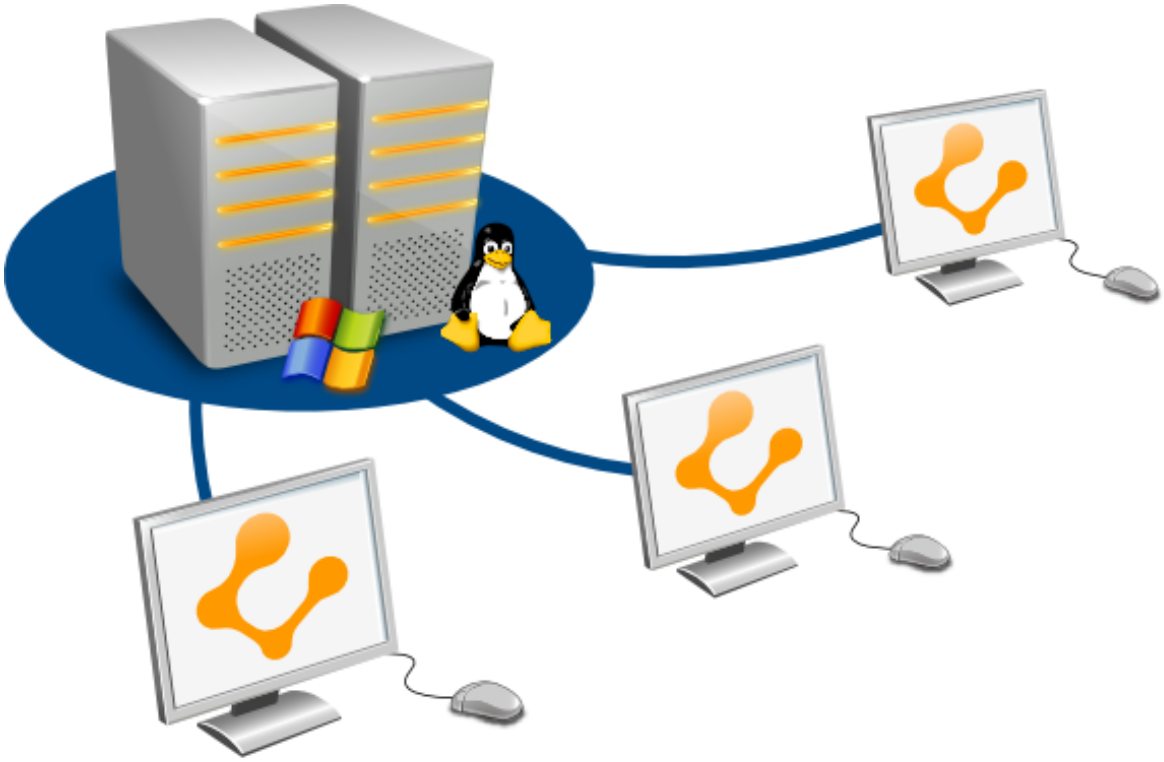
Ulteo Open Virtual Desktop

Protocol Description



Contents

The purpose of this documentation is to describe the protocols used by *Ulteo Open Virtual Desktop*.

1 List of Protocols used

1.1 Hyper Text Transfert Protocol (HTTP)

The base communication protocol used is HTTP because Ulteo is a Web based solution. But we also use HTTP for the communication between servers: Session Manager to Application Server and vice versa. This protocol runs over TCP on the port 80. We are not using HTTPS (443) at the moment.

The Administration Console is web-based too.

HTTP on Wikipedia

1.2 Remote Frame Buffer (RFB)

RFB is the protocol used to display the remote desktop and applications within the Web broswer. This protocol is used by VNC. VNC runs over TCP too. A VNC server runs one instance by desktop so one port for each display. The classic port used by VNC is 5900.

For Ulteo Open Virtual Desktop, we are using ports between 5900 and 6000.

RFB on Wikipedia

1.3 Secure Shell (SSH)

SSH is a protocol used to access a machine securely over a network. SSH uses encryption to make sure that no one on the network can decrypt what the user does.

We are using SSH to tunnel the RFB stream because RFB is not encprypted.

SSH on Wikipedia

1.4 Remote Desktop Protocol (RDP)

RDP is the protocol used by Microsoft Corp.® to display the remote desktop for their Terminal Services™ software.

RDP is used by Ulteo Open Virtual Desktop to display Windows™ applications.

RDP on Wikipedia

2 Server Communication

Servers communicate by using HTTP webservices.

The Session Manager identifies an Application Server using its Fully Qualified Domain Name (FQDN).

Applications Servers only answer to the Session Manager for which the address is stored in a configuration file. (*SESSION_MANAGER_URL*).

2.1 Security

At the moment, the server are authenticated using the DNS resolution system.

When an Application Server sends a status, it sends an extra argument named *fqdn*. The Session Manager performs 2 authentication tests and 1 authorization.

- **FQDN resolution:** resolves the FQDN to get an IP address and tests if it matches the remote server IP ($SERVER['REMOTE_ADDR'] in PHP). Authentication directly depends on that match
- **reverse resolution:** resolves the server IP address and tests if it matches the FQDN argument. Authentication directly depends on that match. This test can be disabled in the administration console using *Disable FQDN check*
- **authorization:** tests the match of the FQDN with one of the *Authorized FQDN* defined in the admin console.

2.2 Windows servers

The Windows agent is using an embeded HTTP server instead of using Apache (as on Linux servers) or IIS™ which is often installed by default on Windows Server 2003™.

As IIS™ is often installed, the Windows agent doesn't use the standart HTTP port but the **8082** port. Another port can be set in the agent configuration file.

2.3 HTTP return codes

Webservices are using the standard HTTP return codes to know if the request succeeded.

- **200 OK :** request succeeded
- **400 Bad Request :** request argument no valid
- **401 Unauthorized :** From SM to ApS: ApS detects that the remote address is not its SM. From ApS to SM: ApS is not registered yet or has invalid authentication.
- **500 Internal Server Error :** Either a bug was detected on the system or the system is broken.

2.4 Session Manager webservices

The webservices used for the servers communication are in */webservices*.

2.4.1 server_status.php

- **method:** GET
- **arguments:**
  - fqdn
  - status: ready/down/broken
- **returns:** GET

An application server sends his status at startup, when it stops or when an important error occured. That's the way the system can know when an offline server is online again or when a server needs to be registered.

2.4.2 server_monitoring.php

- **method:** POST
- **arguments:**
  - fqdn
  - xml: the monitoring XML file
- **returns:** nothing

Application Servers send periodically their system information so the status of each server is known. This is used for the session load balancing for instance.

The monitoring contains the CPU and RAM load. But also running applications for each sessions.

2.4.3 session_invite.php

- **method:** GET
- **arguments:**
  - fqdn
  - session: session id
  - email: the email address to send the invitation
  - mode: active | passive
- **returns:** the token of this share in XML content

Application Servers send a request when a user want create invite someone.

2.4.4 session_status.php

- **method:** GET
- **arguments:**
  - fqdn
  - session: session id
  - status: numerical status
- **returns:** nothing

Application Servers send, for each session, their status each time it changes.

2.4.5 session_token.php

- **method:** GET
- **arguments:**
  - fqdn
  - token: a random string
- **returns:** session information as XML

When a user is launching a session, he is redirected with a token to an Application Server. To verify the user identity and get the session information, the Application Server sending this token to to the Session Manager. If it's valid, it gets information as XML content.

2.4.6 icon.php

Used for Windows applications

- **method:** GET
- **arguments:**
  - fqdn
  - id: an application id
- **returns:** icon image in PNG

Gets the icon for an application so it can display it on the user desktop. The application server uses a cache system so an application icon is only asked once by a server.

2.4.7 application.php

Used for Virtual and Windows applications

- **method:** GET
- **arguments:**
  - fqdn
  - id: the application id
- **returns:** application informations as XML content

Gets the application information from the id argument to create the application shortcut on the user's desktop.

2.5 Application Server webservices

The webservices used for servers communication are in */webservices*.

2.5.1 server_status.php

- **method:** GET
- **arguments:** No
- **returns:** ready/down/broken

The Session Manager asks for the server status periodically to update its status, and detect down or broken states.

2.5.2 session_status.php

- **method:** GET
- **arguments:**
  - session: a session id
- **returns:** numerical session status

The Session Manager asks for each session status periodically to detect servers/sessions craches.

2.5.3 server_monotoring.php

- **method:** GET
- **arguments:** No
- **returns:** monitoring as XML content

Session Manager asks for the server monitoring when it detects too old cache information.

2.5.4 applications.php

- **method:** GET
- **arguments:** No
- **returns:** applications list as XML content

Gets installed applications on the system.

2.5.5 kill_session.php

- **method:** GET
- **arguments:**
  - session: a session id
- **returns:** nothing

Force a specific session to exit without notifying the user.

2.5.6 server_version.php

- **method:** GET
- **arguments:** No
- **returns:** version as plain/text content

Get the server version. On Windows, gets the Windows long version. On Linux, get the content of */etc/issue*.

2.5.7 server_type.php

- **method:** GET
- **arguments:** No
- **returns:** windows/linux

Gets the server type.

2.5.8 icon.php

- **method:** GET
- **arguments:**
  - path: a file path
- **returns:** a PNG image

Extracts the icon associated with an application.

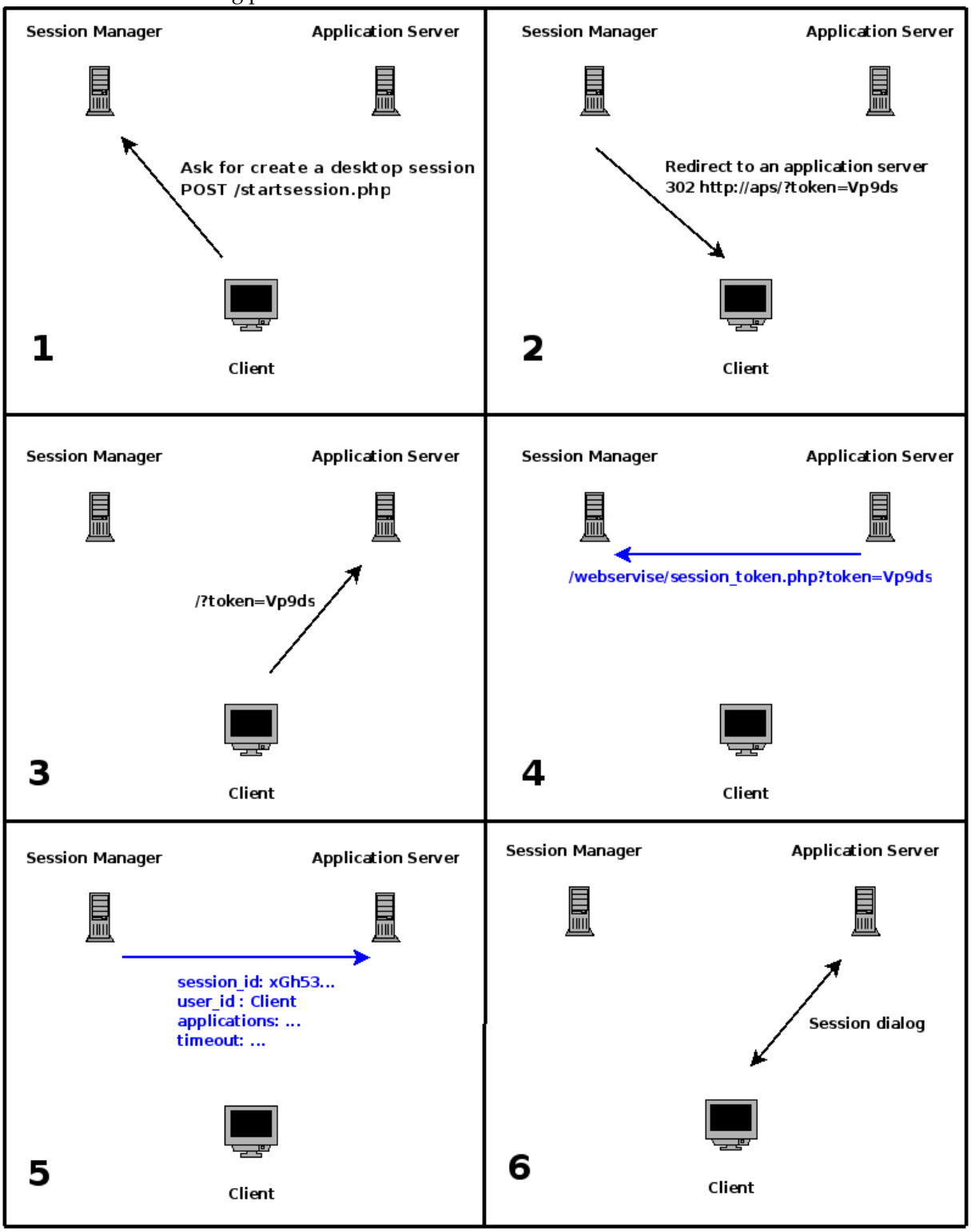
2.5.9 apt-get.php

- **method:** GET
- **arguments:**
  - path: request/status/show
  - request: an apt-get request
  - job: a job id
  - shows: status/stderr/stdout
- **returns:** depends on action type
  - *request:* job id
  - *show:* plain/text content
  - *job:* a job id
  - *status:* -3/-2/-1/0/1/2

Used to install/remove applications.
Gets job id from action request.

3 Desktop Session

Here is the session launching process:



3.1 Client/Server communication

Once the client is redirected to the Application Server, it only requires this server, not the Session Manager, excepted for session sharing as described below.

The session display is encapsulated in a SSH tunnel but webservices are also used to communicate.

For instance, for printing.

3.1.1 Application Server first page informations

the server provides some informations about the session. The used content type is XML. Here is the XML used format:

The root node is called **session** and contains 3 attributes:

- **shareable**: *true* | *false*. Indicate if the session is shareable or not.
- **persistent**: *true* | *false*. Indicate if the session is persitent or not.
- **mode**: *desktop* | *portal*. Indicate if the session is in remote desktop mode or in portal mode.

The root node also contains a child node called **aps** which have 4 attributes:

- **protocol**: *http* | *https*
- **server**: the Application Server hostname
- **port**: the Application Server port of the service
- **location**: the location of the Ulteo part on the HTTP server

Here is an example:

```
<?xml version="1.0" encoding="utf-8"?>
<session mode="desktop" shareable="true" persistent="false">
  <aps protocol="http" server="aps.ulteo.com" port="80" location="/ ↩
     applicationserver" />
</session>
```

Thanks to the previous informations, it's possible to build the base url to call the other webservices like that: ***protocol://server:portlocation***.

3.1.2 access.php

- **method:** GET
- **arguments:**
  - application_id: desktop | portal | application_pid
    An application_pid is an instance of a running application
- **returns:** the session access as XML content

When *whatsup.php* give a session ready, the client has to call this one to get all the access to join the session: SSH port/login/password, VNC port/password ...

XML format:

- **session** Root node
  - **parameters** Node
    - *height* Attribute
    - *width* Attribute
    - *share_desktop* Attribute value: (yes | no)
    - *view_only* Attribute value: (yes | no)
  - **ssh** Node
    - *host* Attribute
    - *passwd* Attribute
    - *user* Attribute
    - *ports* Node
      - *port* Text node

  - **vnc** Node *Optional* (only exist if application_id parameters
    * *passwd* Attribute
    * *port* Attribute
    * *user* Attribute
    * *quality* Node
      · *compression_level* Attribute value: [0-9]
      · *encoding* Attribute value: Tight
      · *jpeg_image_quality* Attribute value: [0-9]
      · *restricted_colors* Attribute value: (yes | no)

Here is an example:

```
<?xml version="1.0" encoding="utf-8"?>
<session>
  <parameters width="1674" height="920" share_desktop="true" view_only="No"/>
  <ssh host="aps.ulteo.com" user="SSH3" passwd="3064396264626635">
    <port>443</port>
    <port>993</port>
    <port>995</port>
  </ssh>
  <vnc port="5901" passwd="43bc4beb567bdaec">
    <quality compression_level="1" restricted_colors="no" jpeg_image_quality="9"  ↩
       encoding="Tight"/>
  </vnc>
</session>
```

3.1.3 apps.php

- **method:** GET
- **arguments:** NONE
- **returns:** an applications list as XML content

The webservice provide the list of available applications for the current session.
The XML root node is *applications* and only contains child nodes called *application*.
An application node only contains attributes:

- *id:* the application identifier (useful to launch the application or get the icon for instance)
- *name:* the pretty name of the application

Here is an example:

```
<?xml version="1.0" encoding="utf-8"?>
<applications>
  <application id="8" name="Firefox Web Browser"/>
  <application id="9" name="GIMP Image Editor"/>
</applications>
```

3.1.4 exit.php

- **method:** ANY
- **arguments:** NONE
- **returns:** bye

This webservice can be called when users close the session window to force the session to exit instead of waiting the timeout trigger.

This is useful on session sharing for instance to be sure that when the client exits, all users sharing the session cannot still access it.

3.1.5 icon.php

- **method:** GET
- **arguments:**
  - id: an application identifier
- **returns:** the application icon as image/png content-type

3.1.6 invite.php

- **method:** POST
- **arguments:**
  - access_id: desktop | application_pid
    An application_pid is an instance of a running application
  - email: the email address to send the invitation
  - mode: active | passive
- **returns:** OK

invite.php is called when the session owner wants to invite someone. The Application Server ask to the Session Manager to create the invitation and send the email.

The *passive* mode is a view of the session where you cannot move the mouse cursor or type keys on your keyboard. It's a view only mode.

3.1.7 print.php

- **method:** GET
- **arguments:**
  - timestamp: a time refered to the pdf file time to download
- **returns:** a PDF file

This webserice is the way to transport the print jobs from the session to the client.

On ApS servers, there is a PDF printer that spools files in the session directory. When there is a new file, *whatsup.php* detects it, extracts the timestamp and alerts the client. Then the client has just to request a print with the timestamp to get the print job and use a mean to print localy (printing applet).

3.1.8 start.php

- **method:** GET | POST
- **arguments:**
  - height: the desired height of the session
  - width: the desired width of the session
- **returns:** nothing

This webservice is called when *whatsup* returned 0 as session status. Startsession sets the session status to 1.

3.1.9 start_app.php

- **method:** GET
- **arguments:**
  - app_id: an application identifier
  - doc: **Optional** an extra argument to give to the application (open a file for instance)
  - size: the desired size of the session ([width]x[height])
- **returns:** either the application instance or an error. Both of them in XML.

The XML root node is *error* the request didn't work.
If the command succeed, the XML root node is *access* and contains one attribute *id* that is the application instance.
This webservice can take some time to answer (maximum 20 seconds)
Here is an example:

```
<?xml version="1.0" encoding="utf-8"?>
<access id="gkFwi"/>
```

3.1.10 suspend.php

- **method:** GET | POST
- **arguments:** NONE
- **returns:** nothing

This webservice is called to order a suspend of the current session.
This is available only if the session is **persistent**.

3.1.11 whatsup.php

- **method:** GET
- **arguments:**
  - application_id: **Optional** a running application instance
- **returns:** session informations as XML content

The webservice has several uses. First of all, it's used to know that the user is still connected (a kind of ping timeout process). Secondly, it allows the client to know session status changes. Finally, it is used to know when a new pdf file is ready for download and print.
The client calls this webservice every 10 seconds top.
XML format:

- **session** Root node
  - **status** Attribute
  - **applications** Node
    - *running* Node (multiple)
      - *app_id* Attribute
      - *job* Attribute
      - *status* Attribute
  - **sharing** Node
    - *count* Attribute
    - *share* Node (multiple)
      - *alive* Attribute (1 | 0)
      - *email* Attribute

· *mode* Attribute (active | passive)

Here is an example:

```
<?xml version="1.0" encoding="utf-8"?>
<session status="2">
  <applications>
    <running job="gKjKg" app_id="15" status="2"/>
    <running job="trV5v" app_id="9" status="2"/>
  </applications>
  <sharing count="1">
    <share email="someone@ulteo.com" mode="passive" joined="1" alive="1"/>
  </sharing>
</session>
```

3.2 Startsession

The main page of the Session Manager is not a needed step to launch a session, it's just an interface to call **startsession.php**.

The goal of this section is to desbribe the API of startsession.php in order to call it using an external HTML file.

- **method:** POST
- **needed arguments:**
  - login
  - password
- **optionnal arguments:**
  - app_with_desktop: 0 | 1 Launch a full desktop when start_app is used
  - client: browser | strict. Define the client type. Default is strict
  - debug: 0 | 1
  - desktop_icons: 0 | 1 Show application icons on the desktop
  - desktop_timeout: a number of second
  - language: fr_FR.UTF-8 | en_GB.UTF-8 | en_US.UTF-8... Desired language and locale for the session (in Unix format)
  - quality: lowest | medium | high | highest
  - session_mode: desktop | portal | external Request for a desktop session, a portal session, or to launch an application from an external web site/web portal
  - size: WIDTHxHEIGHT Specify the size of the session
  - start_app: 1 | 2... An application id to launch at session start (required for "external" mode)
  - start_app_args: /some/file/path... A document path to open inside the OVD (optional)
  - timezone: US/Eastern | Europe/London | Asia/Tokyo... Desired timezone for the session. Default is server timezone
  - windows_keymap: en | fr | ... Force the keymap for Windows applications
- **return:**
  - if fails: a HTML message
  - if success: a HTTP redirection (302) to an application server

Here is a example of a basic HTML page which start a session:

```
<html>
  <body>
<form action="http://sm.ulteo.com/sessionmanager/startsession.php" method="post">
 <input type="hidden" name="client" value="browser" />

 Login:
 <input type="text" name="login" value="" />
 <br/>

 Password:
 <input type="password" name="password" value="" />
 <br/>

 Language:
 <select name="locale">
  <option value="en_GB.UTF-8">English</option>
  <option value="fr_FR.UTF-8">French</option>
 </select>
 <br/>

 <input type="submit" value="Launch" />

</form>
 </body>
</html>
```

4 Numerical ID matching

4.1 Session status

The status sent by webservices and used by both systems (Session Manager and Application Server) are integer figures.

- **-1:** session to create
- **0:** session created
- **1:** session to start
- **22:** session is starting
- **2:** session ready
- **3:** session to destroy
- **4:** unknown session/session destroyed
- **9:** session to suspend
- **10:** session suspended
- **11:** session to restore

4.2 Jobs status

When installing/removing applications using the *apt-get* webservice, the Administration console requests the ApS server to return the status of the job.

- **-3:** server error
- **-2:** job made an error, system can switch in broken mode

- **-1:** job id not exist
- **0:** job is spooling
- **1:** job in progress
- **2:** job finished