[10.2.1](https://www.rfc-editor.org/rfc/rfc3261#section-10.2.1) Adding Bindings

* The REGISTER request sent to a registrar includes the contact address(es) to which SIP requests for the address-of-record should be forwarded. The address-of-record is included in the To header field of the REGISTER request .
* The Contact header field values of the request typically consist of SIP or SIPS URIs that identify particular SIP endpoints
* A SIP UA can choose to register telephone numbers (with the tel URL, [RFC 2806](https://www.rfc-editor.org/rfc/rfc2806) [[9](https://www.rfc-editor.org/rfc/rfc3261#_blank)]) or email addresses (with a mailto URL, [RFC 2368](https://www.rfc-editor.org/rfc/rfc2368) [[32](https://www.rfc-editor.org/rfc/rfc3261#_blank)]) as Contacts for an address-of-record, for example .
* Ngay sau khi đã thiết lập binding, người dùng hoàn toàn có thể sửa đổi. Sau đó bản tin phản hồi 2xx sẽ trả về, chứa đầy đủ các thông tin mà người dùng đã đăng kí.
* If the address-of-record in the To header field of a REGISTER request is a SIPS URI, then any Contact header field values in the request SHOULD also be SIPS URIs. Clients should only register non-SIPS URIs under a SIPS address-of-record when the security of the resource represented by the contact address is guaranteed by other means .

Registrations do not need to update all bindings. Typically, a UA only updates its own contact addresses .

[10.2.1.1](https://www.rfc-editor.org/rfc/rfc3261#section-10.2.1.1) Setting the Expiration Interval of Contact Addresses

* When a client sends a REGISTER request, it MAY suggest an expiration interval.
* There are two ways in which a client can suggest an expiration interval for a binding:
  + Through an Expires header field or an "expires" Contact header parameter.
  + The latter allows expiration intervals to be suggested on a per-binding basis when more than one binding is given in a single REGISTER request, whereas the former suggests an expiration interval for all Contact header field values that do not contain the "expires" parameter .  
    => Nếu client không thể, phía sever có thể sẽ đảm nhận vai trò này.

[10.2.1.2](https://www.rfc-editor.org/rfc/rfc3261#section-10.2.1.2) Preferences among Contact Addresses

* Trong trường hợp có nhiều Contact trong một yêu cẩu REGISTER = > UA sẽ liên kết => Contact header field với address of record trong trường To.
* Để phân biệt, sử dụng tham số q trong Contact header so sánh với các address-of-record.

[10.2.2](https://www.rfc-editor.org/rfc/rfc3261#section-10.2.2) Removing Bindings

* Registrations are soft state and expire unless refreshed, but can also be explicitly removed.
* A UA requests the immediate removal of a binding by specifying an expiration interval of "0" for that contact address in a REGISTER request. UAs SHOULD support this mechanism so that bindings can be removed before their expiration interval has passed .
* The REGISTER-specific Contact header field value of "\*" applies to all registrations, but it MUST NOT be used unless the Expires header field is present with a value of "0" .
* Use of the "\*" Contact header field value allows a registering UA to remove all bindings associated with an address-of-record without knowing their precise values .

[10.2.3](https://www.rfc-editor.org/rfc/rfc3261#section-10.2.3) Fetching Bindings

* Một response thành công tương ứng tới bản tin REGISTER sẽ chứa đầy đủ danh sách các bindings đã setup
* If no Contact header field is present in a REGISTER request, the list of bindings is left unchanged .

[10.2.4](https://www.rfc-editor.org/rfc/rfc3261#section-10.2.4) Refreshing Bindings

* Mỗi UA có trách nghiệm với bindings mà nó tạo, không nên can thiệp của UA khác.
* Bản tin 200OK do registrar quản lý sẽ phản hồi trở lại với đầy đủ các bindings, người dùng sẽ đối chiếu và chọn bindings mà nó mong muốn do tạo mới. Đồng thời mở rộng thêm Expire. Với khoảng thời gian Expire, nó sẽ bắt đầu đăng kí.
* A UA SHOULD use the same Call-ID for all registrations during a single boot cycle. Registration refreshes SHOULD be sent to the same network address as the original registration, unless redirected .

[10.2.5](https://www.rfc-editor.org/rfc/rfc3261#section-10.2.5) Setting the Internal Clock

* Người dùng phải setup theo thời Date header field trong trường hợp bản tin response đề cập.

[10.2.6](https://www.rfc-editor.org/rfc/rfc3261#section-10.2.6) Discovering a Registrar

* UAs can use three ways to determine the address to which to send registrations:
  + by configuration, using the address-of-record, and multicast. A UA can be configured, in ways beyond the scope of this specification, with a registrar address.
  + If there is no configured registrar address, the UA SHOULD use the host part of the address- of-record as the Request-URI and address the request there, using the normal SIP server location mechanisms.
  + Finally, a UA can be configured to use multicast. Multicast registrations are addressed to the well-known "all SIP servers" multicast address "sip.mcast.net" (224.0.1.75 for IPv4).

[10.2.7](https://www.rfc-editor.org/rfc/rfc3261#section-10.2.7) Transmitting a Request

* Sau khi bản tin response được chấp nhận, UAC sẽ thực hiện thủ tục 8.1.2 để chuyển bản tin tới transaction layer.
* If the transaction layer returns a timeout error because the REGISTER yielded no response, the UAC SHOULD NOT immediately re-attempt a registration to the same registrar .
* Waiting some reasonable time interval for the conditions causing the timeout to be corrected reduces unnecessary load on the network

[10.2.8](https://www.rfc-editor.org/rfc/rfc3261#section-10.2.8) Error Responses

* If a UA receives a 423 (Interval Too Brief) response, it MAY retry the registration after making the expiration interval of all contact addresses in the REGISTER request equal to or greater than the expiration interval within the Min-Expires header field of the 423 (Interval Too Brief) response .

[10.3](https://www.rfc-editor.org/rfc/rfc3261#section-10.3) Processing REGISTER Requests

* A registrar is a UAS that responds to REGISTER requests and maintains a list of bindings that are accessible to proxy servers and redirect servers within its administrative domain. A registrar MUST not generate 6xx responses .
* A registrar MAY redirect REGISTER requests as appropriate. One common usage would be for a registrar listening on a multicast interface to redirect multicast REGISTER requests to its own unicast interface with a 302 (Moved Temporarily) response .
* Registrars MUST ignore the Record-Route header field if it is included in a REGISTER request.
* Không bao gồm trường này trong bất kì bản tin phản hồi nào cho REGISTER

A registrar might receive a request that traversed a proxy which treats REGISTER as an unknown request and which added a Record- Route header field value .

When receiving a REGISTER request, a registrar follows these steps:

1. The registrar inspects the Request-URI to determine whether it has access to bindings for the domain identified in the Request-URI. If not, and if the server also acts as a proxy server, the server SHOULD forward the request to the addressed domain, following the general behavior for proxying messages described in [Section 16](https://www.rfc-editor.org/rfc/rfc3261#section-16) .

2. To guarantee that the registrar supports any necessary extensions, the registrar MUST process the Require header field values as described for UASs in [Section 8.2.2](https://www.rfc-editor.org/rfc/rfc3261#section-8.2.2) .

3. A registrar SHOULD authenticate the UAC. Mechanisms for the authentication of SIP user agents are described in [Section 22](https://www.rfc-editor.org/rfc/rfc3261#section-22) .

Registration behavior in no way overrides the generic authentication framework for SIP. If no authentication mechanism is available, the registrar MAY take the From address as the asserted identity of the originator of the request .

4. The registrar SHOULD determine if the authenticated user is authorized to modify registrations for this address-of-record .

5. The registrar extracts the address-of-record from the To header field of the request. If the address-of-record is not valid for the domain in the Request-URI, the registrar MUST send a 404 (Not Found) response and skip the remaining steps. The URI MUST then be converted to a canonical form. To do that, all URI parameters MUST be removed (including the user-param), and any escaped characters MUST be converted to their unescaped form. The result serves as an index into the list of bindings .

6. The registrar checks whether the request contains the Contact header field. If not, it skips to the last step. If the Contact header field is present, the registrar checks if there is one Contact field value that contains the special value "\*" and an Expires field. If the request has additional Contact fields or an expiration time other than zero, the request is invalid, and the server MUST return a 400 (Invalid Request) and skip the remaining steps. If not, the registrar checks whether the Call-ID agrees with the value stored for each binding. If not, it MUST remove the binding. If it does agree, it MUST remove the binding only if the CSeq in the request is higher than the value stored for that binding. Otherwise, the update MUST be aborted and the request fails .

7. The registrar now processes each contact address in the Contact header field in turn. For each address, it determines the expiration interval as follows: - If the field value has an "expires" parameter, that value MUST be taken as the requested expiration .

8. The registrar returns a 200 (OK) response. The response MUST contain Contact header field values enumerating all current bindings. Each Contact value MUST feature an "expires" parameter indicating its expiration interval chosen by the registrar. The response SHOULD include a Date header field .

# [11](https://www.rfc-editor.org/rfc/rfc3261#section-11) Querying for Capabilities

* The SIP method OPTIONS allows a UA to query another UA or a proxy server as to its capabilities. This allows a client to discover information about the supported methods, content types, extensions, codecs, etc. without "ringing" the other party. All UAs MUST support the OPTIONS method .
* The target of the OPTIONS request is identified by the Request-URI, which could identify another UA or a SIP server.
* A server receiving an OPTIONS request with a Max- Forwards header field value of 0 MAY respond to the request regardless of the Request-URI .
* Hành xử tương tự như HTTP/1.1. Sử dụng “traceroute”
* Trong trường hợp transaction layer trả timeout -> OPTIONS không phản hổi -> Không tìm thấy, mất.
* Bản tin yêu cầu OPTIONS có thể một phần sử dụng để thành lập dialog trong tương lai

[11.1](https://www.rfc-editor.org/rfc/rfc3261#section-11.1) Construction of OPTIONS Request

* Cấu trúc quy tắc sử dụng như trong phần 8.1.1
* A Contact header field MAY be present in an OPTIONS .
* Cần lưu ý việc cho body SDP để cung cấp thêm thông tin ma UA mong muốn.
* The response to an OPTIONS request is assumed to be scoped to the Request-URI in the original request.

[11.2](https://www.rfc-editor.org/rfc/rfc3261#section-11.2) Processing of OPTIONS Request

* The response to an OPTIONS is constructed using the standard rules for a SIP response as discussed in [Section 8.2.6](https://www.rfc-editor.org/rfc/rfc3261#section-8.2.6).
* The response code chosen MUST be the same that would have been chosen had the request been an INVITE. That is, a 200 (OK) would be returned if the UAS is ready to accept a call, a 486 (Busy Here) would be returned if the UAS is busy, etc.
* Xác định được trạng thái UAS
* An OPTIONS request received within a dialog generates a 200 (OK) response => Không ảnh hưởng tới các trao đổi luồng khác.
* This use of OPTIONS has limitations due to the differences in proxy handling of OPTIONS and INVITE requests.
* While a forked INVITE can result in multiple 200 (OK) responses being returned, a forked OPTIONS will only result in a single 200 (OK) response,
* If the response to an OPTIONS is generated by a proxy server, the proxy returns a 200 (OK), listing the capabilities of the server .
* The response does not contain a message body .
* Allow, Accept, Accept-Encoding, Accept-Language, and Supported header fields SHOULD be present in a 200 (OK) response to an OPTIONS request.
* If the response is generated by a proxy, the Allow header field SHOULD be omitted as it is ambiguous since a proxy is method agnostic. Contact header fields MAY be present in a 200 (OK) response and have the same semantics as in a 3xx response. That is, they may list a set of alternative names and methods of reaching the user. A Warning header field MAY be present .

Example OPTIONS response generated by a UAS (corresponding to the request in [Section 11.1](https://www.rfc-editor.org/rfc/rfc3261#section-11.1)): SIP/2.0 200 OK Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKhjhs8ass877 ;received=192.0.2.4 To: <sip:carol@chicago.com>;tag=93810874 From: Alice <sip:alice@atlanta.com>;tag=1928301774 Call-ID: a84b4c76e66710 CSeq: 63104 OPTIONS Contact: <sip:carol@chicago.com> Contact: <mailto:carol@chicago.com> Allow: INVITE, ACK, CANCEL, OPTIONS, BYE Accept: application/sdp Accept-Encoding: gzip Accept-Language: en Supported: foo Content-Type: application/sdp Content-Length: 274 (SDP not shown)

[**12**](https://www.rfc-editor.org/rfc/rfc3261#section-12) **Dialogs**

* A dialog represents a peer-to-peer SIP relationship between two user agents that persists for some time. The dialog facilitates sequencing of messages between the user agents and proper routing of requests between both of them. The dialog represents a context in which to interpret SIP messages. .
* A dialog is identified at each UA with a dialog ID, which consists of a Call-ID value, a local tag and a remote tag. The dialog ID at each UA involved in the dialog is not the same. Specifically, the local tag at one UA is identical to the remote tag at the peer UA. The tags are opaque tokens that facilitate the generation of unique dialog IDs .
* A dialog ID is also associated with all responses and with any request that contains a tag in the To field. The rules for computing the dialog ID of a message depend on whether the SIP element is a UAC or UAS.
* For a UAC, the Call-ID value of the dialog ID is set to the Call-ID of the message, the remote tag is set to the tag in the To field of the message, and the local tag is set to the tag in the From field of the message (these rules apply to both requests and responses). As one would expect for a UAS, the Call-ID value of the dialog ID is set to the Call-ID of the message, the remote tag is set to the tag in the From field of the message, and the local tag is set to the tag in the To field of the message .

A dialog contains certain pieces of state needed for further message transmissions within the dialog. This state consists of the dialog ID, a local sequence number (used to order requests from the UA to its peer), a remote sequence number (used to order requests from its peer to the UA), a local URI, a remote URI, remote target, a boolean flag called "secure", and a route set, which is an ordered list of URIs. The route set is the list of servers that need to be traversed to send a request to the peer. A dialog can also be in the "early" state, which occurs when it is created with a provisional response, and then transition to the "confirmed" state when a 2xx final response arrives. For other responses, or if no response arrives at all on that dialog, the early dialog terminates .

[12.1](https://www.rfc-editor.org/rfc/rfc3261#section-12.1) Creation of a Dialog

* Dialogs are created through the generation of non-failure responses to requests with specific methods. Within this specification, only 2xx and 101-199 responses with a To tag, where the request was INVITE, will establish a dialog. A dialog established by a non-final response to a request is in the "early" state and it is called an early dialog. Extensions MAY define other means for creating dialogs UAs MUST assign values to the dialog ID components as described below .

[12.1.1](https://www.rfc-editor.org/rfc/rfc3261#section-12.1.1) UAS behavior

* When a UAS responds to a request with a response that establishes a dialog (such as a 2xx to INVITE), the UAS MUST copy all Record-Route header field values from the request into the response (including the URIs, URI parameters, and any Record-Route header field parameters, whether they are known or unknown to the UAS) and MUST maintain the order of those values. The UAS MUST add a Contact header field to the response. The Contact header field contains an address where the UAS would like to be contacted for subsequent requests in the dialog (which includes the ACK for a 2xx response in the case of an INVITE) .

Generally, the host portion of this URI is the IP address or FQDN of the host. The URI provided in the Contact header field MUST be a SIP or SIPS URI. If the request that initiated the dialog contained a SIPS URI in the Request-URI or in the top Record-Route header field value, if there was any, or the Contact header field if there was no Record-Route header field, the Contact header field in the response MUST be a SIPS URI. The URI SHOULD have global scope (that is, the same URI can be used in messages outside this dialog). The same way, the scope of the URI in the Contact header field of the INVITE is not limited to this dialog either. It can therefore be used in messages to the UAC even outside this dialog .

* The UAS then constructs the state of the dialog. This state MUST be maintained for the duration of the dialog .
* If the request arrived over TLS, and the Request-URI contained a SIPS URI, the "secure" flag is set to TRUE .
* The route set MUST be set to the list of URIs in the Record-Route header field from the request, taken in order and preserving all URI parameters.
* If no Record-Route header field is present in the request, the route set MUST be set to the empty set. This route set, even if empty, overrides any pre-existing route set for future requests in this dialog. The remote target MUST be set to the URI from the Contact header field of the request .
* The remote sequence number MUST be set to the value of the sequence number in the CSeq header field of the request. The local sequence number MUST be empty.
* The call identifier component of the dialog ID MUST be set to the value of the Call-ID in the request. The local tag component of the dialog ID MUST be set to the tag in the To field in the response to the request (which always includes a tag), and the remote tag component of the dialog ID MUST be set to the tag from the From field in the request. A UAS MUST be prepared to receive a request without a tag in the From field, in which case the tag is considered to have a value of null .
* The remote URI MUST be set to the URI in the From field, and the local URI MUST be set to the URI in the To field .

[12.1.2](https://www.rfc-editor.org/rfc/rfc3261#section-12.1.2) UAC Behavior

* When a UAC sends a request that can establish a dialog (such as an INVITE) it MUST provide a SIP or SIPS URI with global scope (i.e., the same SIP URI can be used in messages outside this dialog) in the Contact header field of the request.
* If the request has a Request- URI or a topmost Route header field value with a SIPS URI, the Contact header field MUST contain a SIPS URI .
* When a UAC receives a response that establishes a dialog, it constructs the state of the dialog. This state MUST be maintained for the duration of the dialog .
* If the request was sent over TLS, and the Request-URI contained a SIPS URI, the "secure" flag is set to TRUE .
* The route set MUST be set to the list of URIs in the Record-Route header field from the response, taken in reverse order and preserving all URI parameters. If no Record-Route header field is present in the response, the route set MUST be set to the empty set. This route set, even if empty, overrides any pre-existing route set for future requests in this dialog. The remote target MUST be set to the URI from the Contact header field of the response .
* The local sequence number MUST be set to the value of the sequence number in the CSeq header field of the request. The remote sequence number MUST be empty (it is established when the remote UA sends a request within the dialog). The call identifier component of the dialog ID MUST be set to the value of the Call-ID in the request .
* The local tag component of the dialog ID MUST be set to the tag in the From field in the request, and the remote tag component of the dialog ID MUST be set to the tag in the To field of the response. A UAC MUST be prepared to receive a response without a tag in the To field, in which case the tag is considered to have a value of null .

This is to maintain backwards compatibility with [RFC 2543](https://www.rfc-editor.org/rfc/rfc2543), which did not mandate To tags .

The remote URI MUST be set to the URI in the To field, and the local URI MUST be set to the URI in the From field .

[12.2](https://www.rfc-editor.org/rfc/rfc3261#section-12.2) Requests within a Dialog

* Once a dialog has been established between two UAs, either of them MAY initiate new transactions as needed within the dialog. The UA sending the request will take the UAC role for the transaction. The UA receiving the request will take the UAS role. Note that these may be different roles than the UAs held during the transaction that established the dialog .
* Requests within a dialog MAY contain Record-Route and Contact header fields. However, these requests do not cause the dialog's route set to be modified, although they may modify the remote target URI .
* Note that an ACK is NOT a target refresh request .
* Target refresh requests only update the dialog's remote target URI, and not the route set formed from the Record-Route. Updating the latter would introduce severe backwards compatibility problems with [RFC 2543](https://www.rfc-editor.org/rfc/rfc2543)-compliant systems .

[12.2.1](https://www.rfc-editor.org/rfc/rfc3261#section-12.2.1) UACBehavior

[12.2.1.1](https://www.rfc-editor.org/rfc/rfc3261#section-12.2.1.1) Generating the Request

A request within a dialog is constructed by using many of the components of the state stored as part of the dialog .

* Trong truờng thông tin To thì URI = remote URI từ dialog state, tag = remote tag của dialog ID, from URI = local URI. Ngoài ra, tag = local tag của dialog ID
* Trong truờng hợp, remote hoặc local tags là null, các tham số tưong ứng sẽ phải loại bỏ
* Usage of the URI from the To and From fields in the original request within subsequent requests is done which used the URI for dialog identification. In this specification, only the tags are used for dialog identification.
* The Call-ID of the request MUST be set to the Call-ID of the dialog .
* Tham số CSeg sẽ tăng liên tục lên 1 qua mỗi hop, ngoại trừ các bản tin ACK và CANCEL. Nếu để trông thì phải tuân theo quy tắc của mục 8.1.1.5
* The UAC uses the remote target and route set to build the Request-URI and Route header field of the request .
* If the route set is empty, the UAC MUST place the remote target URI into the Request-URI. The UAC MUST NOT add a Route header field to the request .
* If the route set is not empty, and the first URI in the route set contains the lr parameter (see [Section 19.1.1](https://www.rfc-editor.org/rfc/rfc3261#section-19.1.1)), the UAC MUST place the remote target URI into the Request-URI and MUST include a Route header field containing the route set values in order, including all parameters .
* If the route set is not empty, and its first URI does not contain the lr parameter, the UAC MUST place the first URI from the route set into the Request-URI, stripping any parameters that are not allowed in a Request-URI. The UAC MUST add a Route header field containing the remainder of the route set values in order, including all parameters. The UAC MUST then place the remote target URI into the Route header field as the last value .
* A UAC SHOULD include a Contact header field in any target refresh requests within a dialog, and unless there is a need to change it, the URI SHOULD be the same as used in previous requests within the dialog. If the "secure" flag is true, that URI MUST be a SIPS URI .
* Một "targer refresh request updates" về remote targer URI -> thay đổi Contact header field -> địa chỉ thay đổi
* Truờng hợp nguợc lại sẽ không tác dụng tới remote targer URI.
* Phần còn lại được miêu tả như mục 8.1.1
* Ngay sau khi yêu cầu đuợc nhận bởi UAS, địa chỉ sever đuợc tính toán, và sau đó sẽ sử dụng thủ tục như mục 8.1.2

[12.2.1.2](https://www.rfc-editor.org/rfc/rfc3261#section-12.2.1.2) Processing the Responses

* The UAC will receive responses to the request from the transaction layer. If the client transaction returns a timeout, this is treated as a 408 (Request Timeout) response .
* Trong truờng hợp UAC nhận đưọc response 3xx, nó sẽ hành xử như mục 8.1.3.4
* Note, however, that when the UAC tries alternative locations, it still uses the route set for the dialog to build the Route header of the request .
* When a UAC receives a 2xx response to a target refresh request, it MUST replace the dialog's remote target URI with the URI from the Contact header field in that response.
* If the response for a request within a dialog is a 481 (Call/Transaction Does Not Exist) or a 408 (Request Timeout), the UAC SHOULD terminate the dialog.
* A UAC SHOULD also terminate a dialog if no response at all is received for the request (the client transaction would inform the TU about the timeout.)

[12.2.2](https://www.rfc-editor.org/rfc/rfc3261#section-12.2.2) UAS Behavior

* Việc cấp phát trạng thái tuỳ thuộc vào UAS có nhận đuợc yêu cầu mong muốn hay không.
* Với mỗi một yêu cầu khác nhau sẽ tác động đến các trạng thái khác nhau.
* The UAS will receive the request from the transaction layer. If the request has a tag in the To header field, the UAS core computes the dialog identifier corresponding to the request and compares it with existing dialogs. If there is a match, this is a mid-dialog request .
* Với truờng hợp cho yêu cầu lần đầu tiên ngoài dialog, nó sẽ thực hiện như mục 8.2
* If the request has a tag in the To header field, but the dialog identifier does not match any existing dialogs, the UAS may have crashed and restarted. Hoặc cũng có thể UAS sẽ tự thích ứng để xác thực
* Another possibility is that the incoming request has been simply misrouted. Based on the To tag, the UAS MAY either accept or reject the request.
* Accepting the request for acceptable To tags provides robustness, so that dialogs can persist even through crashes. UAs wishing to support this capability must take into consideration some issues such as choosing monotonically increasing CSeq sequence numbers even across reboots, reconstructing the route set, and accepting out-of-range RTP timestamps and sequence numbers .
* Trong trường hợp từ chối không muốn khởi tạo lại dialog, UAS sẽ guiwr bản tin 481 ( Call/Transaction Does Not Exist)
* Requests that do not change in any way the state of a dialog may be received within a dialog (for example, an OPTIONS request). They are processed as if they had been received outside the dialog .
* If the remote sequence number is empty, it MUST be set to the value of the sequence number in the CSeq header field value in the request .
* If the remote sequence number was not empty, but the sequence number of the request is lower than the remote sequence number, the request is out of order and MUST be rejected with a 500 (Server Internal Error) response.
* If the remote sequence number was not empty, and the sequence number of the request is greater than the remote sequence number, the request is in order.

=> UAS sẵn sàng thích ứng để tiếp nhận.

* If a proxy challenges a request generated by the UAC, the UAC has to resubmit the request with credentials. The resubmitted request will have a new CSeq number. The UAS will never see the first request, and thus, it will notice a gap in the CSeq number space .

=> Gap không phải bất kì loại lỗi nào cả

* When a UAS receives a target refresh request, it MUST replace the dialog's remote target URI with the URI from the Contact header field in that request.

[12.3](https://www.rfc-editor.org/rfc/rfc3261#section-12.3) Termination of a Dialog

* if a request outside of a dialog generates a non-2xx final response, any early dialogs created through provisional responses to that request are terminated.

=> Method Bye is Ex.

[13](https://www.rfc-editor.org/rfc/rfc3261#section-13) Initiating a Session

[13.1](https://www.rfc-editor.org/rfc/rfc3261#section-13.1) Overview

* Ngưoi dùng mong muốn thành lập session, sẽ guiwr bản tin INVITE request, bản tin này có thể qua nhiều proxys và nhiều UASs. Sau khi chấp nhận bởi UASs nó sẽ gửi bản tin 2xx
* If the invitation is not accepted, a 3xx, 4xx, 5xx or 6xx response is sent, depending on the reason for the rejection. Các bản tin 1xx cũng sẽ đưọc gửi trưóc đó để thông báo tín trình đang được xử lý.
* Once it receives a final response, the UAC needs to send an ACK for every final response it receives. The procedure for sending this ACK depends on the type of response. For final responses between 300 and 699, the ACK processing is done in the transaction layer and follows one set of rules (See [Section 17](https://www.rfc-editor.org/rfc/rfc3261#section-17)). For 2xx responses, the ACK is generated by the UAC core .
* Có truờng hợp tạo nhiều dialog với nhiều UASs, vì INVITE forked.

[13.2](https://www.rfc-editor.org/rfc/rfc3261#section-13.2) UAC Processing

[13.2.1](https://www.rfc-editor.org/rfc/rfc3261#section-13.2.1) Creating the Initial INVITE

* Since the initial INVITE represents a request outside of a dialog, its construction follows the procedures of [Section 8.1.1](https://www.rfc-editor.org/rfc/rfc3261#section-8.1.1).
* Các trường thông tin nên có trong INVITE
* An Allow header field ([Section 20.5](https://www.rfc-editor.org/rfc/rfc3261#section-20.5)) SHOULD be present in the INVITE .It indicates what methods can be invoked within a dialog, on the UA sending the INVITE, for the duration of the dialog.
* A Supported header field ([Section 20.37](https://www.rfc-editor.org/rfc/rfc3261#section-20.37)) SHOULD be present in the INVITE. It enumerates all the extensions understood by the UAC .
* An Accept ([Section 20.1](https://www.rfc-editor.org/rfc/rfc3261#section-20.1)) header field MAY be present in the INVITE .

=> Phục vụ cho đa dạng yêu cầu miêu tả định dạng phiên cho cả ngưòi nhận và nguời gửi.

* The UAC MAY add an Expires header field ([Section 20.19](https://www.rfc-editor.org/rfc/rfc3261#section-20.19)) to limit the validity of the invitation.

=> Nếu hết time, gửi bản tin CACEL => kết thúc phiên

A UAC MAY also find it useful to add, among others, Subject ([Sectio](https://www.rfc-editor.org/rfc/rfc3261#section-20.36) [20.36](https://www.rfc-editor.org/rfc/rfc3261#section-20.36)), Organ[n](https://www.rfc-editor.org/rfc/rfc3261#section-20.36)ization ([Section 20.25](https://www.rfc-editor.org/rfc/rfc3261#section-20.25)) and User-Agent ([Section 20.41](https://www.rfc-editor.org/rfc/rfc3261#section-20.41)) header fields. They all contain information related to the INVITE .

* Thêm thành phần miêu tả phiên SDP, đã đuợc trình bày trước đó.
* SIP uses an offer/answer model => Giới hạn những yêu cầu mà ngưòi dùng SIP mong muốn, đồng thời hạn chế những yêu cầu không đáp ứng khác.
* For the initial INVITE transaction, the rules are:

o The initial offer MUST be in either an INVITE or, if not there, in the first reliable non-failure message from the UAS back to the UAC. In this specification, that is the final 2xx response .

=> Đồng thời xác nhận thứ tự ưu tiên từ bản tin INVITE nó gửi

o If the initial offer is in the first reliable non-failure message from the UAS back to UAC, the answer MUST be in the acknowledgement for that message (in this specification, ACK for a 2xx response) .

o After having sent or received an answer to the first offer, the UAC MAY generate subsequent offers in requests based on rules specified for that method, but only if it has received answers to any previous offers, and has not sent any offers to which it hasn't gotten an answer .

o Once the UAS has sent or received an answer to the initial offer, it MUST NOT generate subsequent offers in any responses to the initial INVITE.

=> Cho đến khi hoàn thành transaction

* SDP and INVITE must be supported by UAs
* The restrictions of the offer-answer model just described only apply to bodies whose Content-Disposition header field value is "session" .
* Therefore, it is possible that both the INVITE and the ACK contain a body message (for example, the INVITE carries a photo (Content- Disposition: render) and the ACK a session description (Content- Disposition: session)) .
* If the Content-Disposition header field is missing, bodies of Content-Type application/sdp imply the disposition "session", while other content types imply "render" .
* Once the INVITE has been created, the UAC follows the procedures defined for sending requests outside of a dialog ([Section 8](https://www.rfc-editor.org/rfc/rfc3261#section-8)).

[13.2.2](https://www.rfc-editor.org/rfc/rfc3261#section-13.2.2) Processing INVITE Responses

Once the INVITE has been passed to the INVITE client transaction, the UAC waits for responses for the INVITE. If the INVITE client transaction returns a timeout rather than a response the TU acts as if a 408 (Request Timeout) response had been received, as described in [Section 8.1.3](https://www.rfc-editor.org/rfc/rfc3261#section-8.1.3) .

[13.2.2.1](https://www.rfc-editor.org/rfc/rfc3261#section-13.2.2.1) 1xx Responses

* Zero, one or multiple provisional responses may arrive before one or more final responses are received. Provisional responses for an INVITE request can create "early dialogs". If a provisional response has a tag in the To field, and if the dialog ID of the response does not match an existing dialog, one is constructed using the procedures defined in [Section](https://www.rfc-editor.org/rfc/rfc3261#section-12.1.2) 12.1.2 .
* The early dialog will only be needed if the UAC needs to send a request to its peer within the dialog before the initial INVITE transaction completes. Header fields present in a provisional response are applicable as long as the dialog is in the early state.

[13.2.2.2](https://www.rfc-editor.org/rfc/rfc3261#section-13.2.2.2) 3xx Responses

* A 3xx response may contain one or more Contact header field values providing new addresses where the callee might be reachable .

[13.2.2.3](https://www.rfc-editor.org/rfc/rfc3261#section-13.2.2.3) 4xx, 5xx and 6xx Responses

* 4xx, 5xx and 6xx responses may contain a Contact header field value indicating the location where additional information about the error can be found.

After having received the non-2xx final response the UAC core considers the INVITE transaction completed. The INVITE client transaction handles the generation of ACKs for the response (see [Section 17](https://www.rfc-editor.org/rfc/rfc3261#section-17)) .

[13.2.2.4](https://www.rfc-editor.org/rfc/rfc3261#section-13.2.2.4) 2xx Responses

* Trong truờng hợp nhận đuợc nhiều phản hồi 2xx từ một INVITE duy nhất gửi đi. Lý do bởi hiện tượng forking proxy. Việc của phía bên nhận phải đối chiếu và chọn lựa với phản hồi thích hợp nhất. Nếu không một new dialog sẽ đuợc thực hiện như mục 12.1.2
* Việc cập nhật trạng thái sẽ diễn ra với từng route set. Các trường hợp trạng thái khác, như số thứ tự (remote and local) sẽ không đuợc tính toán. => Việc update state of dialog sẽ chỉ là hữu hạn
* Khi nhận đuợc response từ phía Sever, cấu trúc phản hồi của bản tin ACK cũng tuơng tự như phần khác, nhưng điểm khác ở đây là phải khớp theo thứ tự phản hồi Cseq, và các thông tin chứng thức phải đi kèm nếu bên kia yêu cầu.
* Once the ACK has been constructed, the procedures of [[4](https://www.rfc-editor.org/rfc/rfc3261#_blank)] are used to determine the destination address, port and transport. However, the request is passed to the transport layer directly for transmission, rather than a client transaction. This is because the UAC core handles retransmissions of the ACK, not the transaction layer. The ACK MUST be passed to the client transport every time a retransmission of the 2xx final response that triggered the ACK arrives .
* Nếu muốn kết thúc dialog sau khi nhận đuợc 2xx response, thì sử dụng cơ chế BYE miêu tả trong mục 15.

[13.3](https://www.rfc-editor.org/rfc/rfc3261#section-13.3) UAS Processing

[13.3.1](https://www.rfc-editor.org/rfc/rfc3261#section-13.3.1) Processing of the INVITE

The UAS core will receive INVITE requests from the transaction layer .

* It first performs the request processing procedures of [Section 8.2](https://www.rfc-editor.org/rfc/rfc3261#section-8.2), which are applied for both requests inside and outside of a dialog .

Assuming these processing states are completed without generating a response, the UAS core performs the additional processing steps:

1. If the request is an INVITE that contains an Expires header field, the UAS core sets a timer for the number of seconds indicated in the header field value. When the timer fires, the invitation is considered to be expired. If the invitation expires before the UAS has generated a final response, a 487 (Request Terminated) response SHOULD be generated .

2. If the request is a mid-dialog request, the method-independent processing described in [Section 12.2.2](https://www.rfc-editor.org/rfc/rfc3261#section-12.2.2) is first applied. It might also modify the session; [Section 14](https://www.rfc-editor.org/rfc/rfc3261#section-14) provides details .

3. If the request has a tag in the To header field but the dialog identifier does not match any of the existing dialogs, the UAS may have crashed and restarted, or may have received a request for a different (possibly failed) UAS. [Section 12.2.2](https://www.rfc-editor.org/rfc/rfc3261#section-12.2.2) provides guidelines to achieve a robust behavior under such a situation .

Processing from here forward assumes that the INVITE is outside of a dialog, and is thus for the purposes of establishing a new session .

* The INVITE may contain a session description, in which case the UAS is being presented with an offer for that session. It is possible that the user is already a participant in that session, even though the INVITE is outside of a dialog. This can happen when a user is invited to the same multicast conference by multiple other participants. If desired, the UAS MAY use identifiers within the session description to detect this duplication. For example, SDP