Remote Function Calls

Everything about RFC

Jays

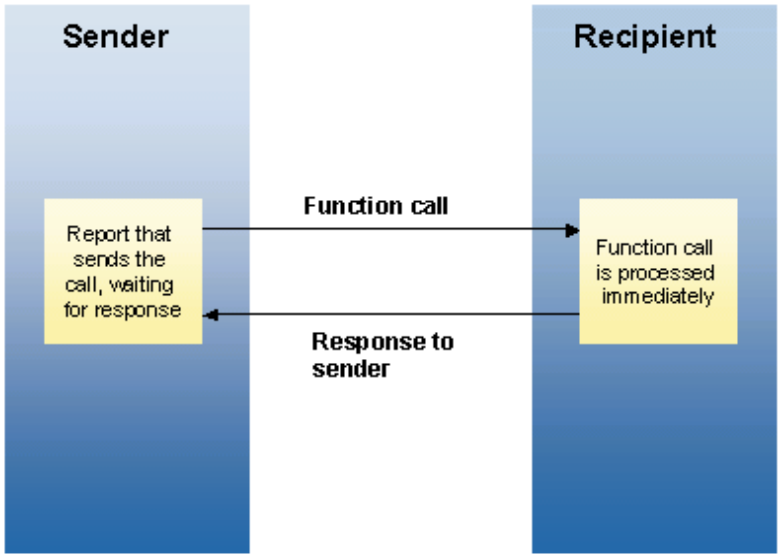
Remote Function Calls

- Communication between applications in different systems in the SAP environment includes connections between SAP systems as well as between SAP systems and non-SAP systems.
- Remote Function Call (RFC) is the standard SAP interface for communication between SAP systems.
- RFC calls a function to be executed in a remote system.
- Although an RFC associated with communication between two different systems, it can also be created for communication within a system.

Communication Types:

- **Synchronous Communication:** *Synchronous* communication uses a *single* function call. Prerequisite for this is that at the time the call is made (or the message is sent), the receiving system is also active and can accept the call and further process it if necessary.
    - **Advantage:** Synchronous communication can be implemented in function calls that require the immediate return of data to the sender system.
    - Example: You create a purchase order with account assignment in the sender system, and you want to perform a budget check in central accounting before you save the purchase order.
    - **Disadvantage:** You need to ensure that both systems are active and can be contacted. If they are not, this can lead to serious disruption of processes. In particular, problems can arise if the receiving system is not available for long periods of time due to maintenance (for example, for a system upgrade).

Synchronous Communication



-

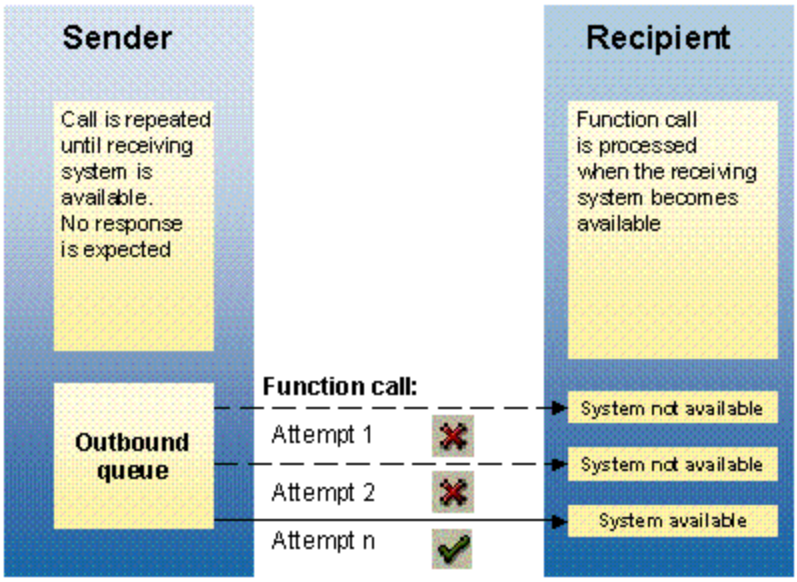
- **Asynchronous Communication:** For *asynchronous* communication, the receiving system does not necessarily have to be available at the time a function call is dispatched from the sender system. The receiving system can receive and process the call at a later time. If the receiving system is not available, the function call remains in the outbound queue of the sending system, from where the call is repeated at regular intervals until it can be processed by the receiving system.

  - **Advantage**: The receiving system does not have to be available at the time the function call is made. If the system is unavailable for a long period of time, for example, for an upgrade, it can still process the data that has been sent in the interim at a later time, and processes in the sending system remain unharmed.

  - Example: You are sending a purchase order to a vendor system. The sending system cannot influence the availability of the receiving system. If the receiving system is not available, the purchase order can be sent repeatedly until the vendor system is available again.

  - **Disadvantage**: Processes that require an immediate response to the sender system cannot be executed using this method.

Asynchronous Communication



Types of RFC:

Based on types of communication:

Synchronous RFC: The first version of RFC is *synchronous* RFC (sRFC). This type of RFC executes the function call based on *synchronous communication*, meaning that the systems involved must both be available at the time the call is made.

- Used for communication between systems
- Used for communication between the SAP application layer and SAP GUI.

Transactional RFC (tRFC)

- *Transactional RFC*(tRFC, previously known as asynchronous RFC) is an ***asynchronous communication*** method that executes the called function module just once in the RFC server. The remote system need not be available at the time when the RFC client program is executing a ***tRFC***. The tRFC component stores the called RFC function, together with the corresponding data, in the SAP database under a unique transaction ID (TID).
- If a call is sent, and the receiving system is down, the call remains in the local queue. The calling dialog program can proceed without waiting to see whether the remote call was

successful. If the receiving system does not become active within a certain amount of time, the call is scheduled to run in batch.

- tRFC is always used if a function is executed as a ***Logical Unit of Work*** (LUW).
- **Disadvantages of tRFC**
- tRFC processes all LUWs independently of one another. Due to the amount of activated tRFC processes, this procedure can reduce performance significantly in both the send and the target systems.
- In addition, the sequence of LUWs defined in the application cannot be kept. It is therefore impossible to guarantee that the transactions will be executed in the sequence dictated by the application. The only thing that can be guaranteed is that all LUWs are transferred sooner or later.

Queued RFC (qRFC)

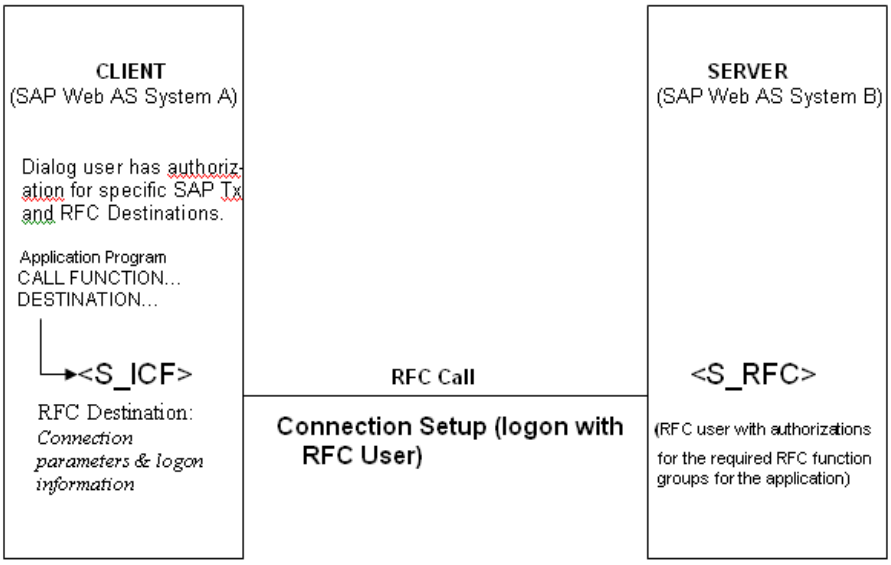
- To guarantee that multiple LUWs are processed in the order specified by the application, tRFC can be serialized using queues (inbound and outbound queues). This type of RFC is called queued RFC (qRFC).
- qRFC is therefore an extension of tRFC. It transfers an LUW (transaction) only if it has no predecessors (based on the sequence defined in different application programs) in the participating queues.
- Implementation of qRFC is recommended if you want to guarantee that several transactions are processed in a predefined order.
- In ABAP coding, transactional RFCs are called via statement Call FUNCTION ... DESTINATION ...IN BACKGROUND TASK.
- Do not get confused! Despite the wording IN BACKGROUND TASK, the tRFC is executed in a dialog process.
- Errors in processing tRFCs can be displayed in transaction SM58.

Restricting Authorizations for RFC Calls:

- *When calling a function module using the RFC interface, the calling program must specify the parameters of the connection in the form of a destination. This destination defines the type of connection, the partner program, and the target system. You can*

*manage it using transaction **SM59**, and it distinguishes between a variety of connections, such as TCP/IP or SAP connections.*

- An RFC destination describes a data record that is stored in the RFC client and contains two types of information: data that describes the network connection, and authentication data for the RFC user. Authentication data is required only if the server system is an SAP Web AS.
- RFC calls can be of two basic types:
- **Untrusted and Trusted.**
- The difference between these two types is that if an untrusted RFC call is made, the client has to authenticate itself to the server using the proper RFC user credentials. If a trusted RFC call is made, no authentication is necessary, as the server system trusts the client.



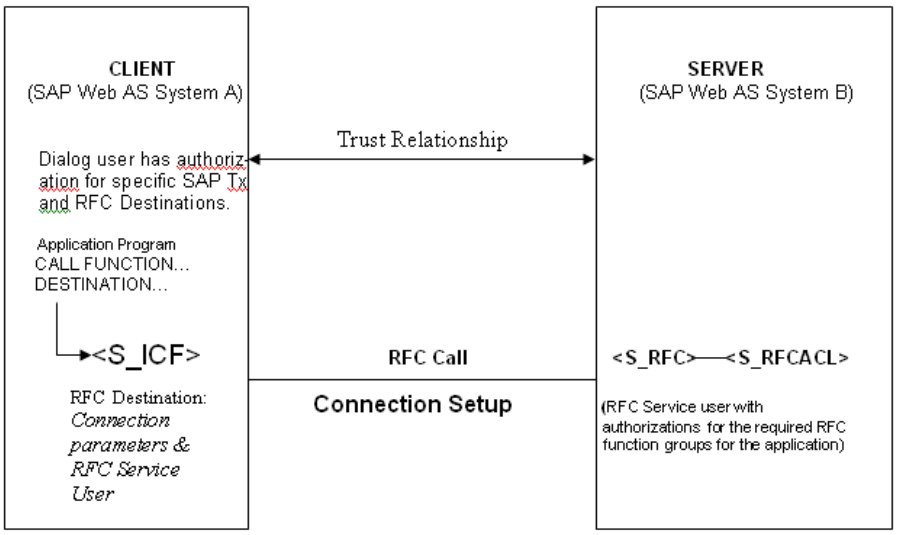
Untrusted RFC Call

- The above figure shows the basic procedure of an untrusted RFC call and the authorization objects that are checked in this procedure. With this type of call, a user in SAP Web AS system A starts a particular transaction in which an RFC call is made to SAP Web AS system B.

- In system A, the authorization object **S_ICF** is checked which contains the **ICF_FIELD** and **ICF_VALUE** authorization fields. The **ICF_FIELD** field can have either of the values "***Service***" or "***Dest***". "***Service***" has to be defined if an Internet Connection Framework service is called. "***Dest***" can be selected if the call is an RFC call, defined in transaction **SM59**. The **ICF_VALUE** field can contain the ICF services or RFC destinations for the user, or better still, for the role. Table **RFCDES** contains a list of the possible RFC destinations.
- To make the RFC call to SAP Web AS system B, the client has to log on using the RFC user that has been defined in SAP Web AS system B for this call. Then, in SAP Web AS system B, the RFC call is granted the authorizations that have been defined for the RFC user in authorization object **S_RFC.**

Authorization object S_RFC has the following authorization fields:

- **RFC_TYPE**
- Type of the RFC object that is to be protected. Currently, this field can take only the value "FUGR" (function group).
- **RFC_NAME**
- Name of the RFC object that is to be protected. The field contain the name of the function groups that can be called by the RFC user. It is important that full authorization – that is , "*****" is not entered here, or else the user would be able to call critical function groups for creating users. Unfortunately, this rule is often not adhered to.
- **ACTVT**
- Activity. Currently this field can take only the value "16" (execute).



Trusted RFC Call

- If an RFC call is made between trusted systems, the call proceeds as follows (as shown in the figure above). There is no need for the clients to log on using an RFC user. Instead, the user ID of the dialog user that is active in SAP Web AS system A is transferred. In this case, the dialog user becomes an RFC service user. For this reason, it is important that there is an additional authorization check in SAP Web AS system B, using authorization object **S_RFCACL.** This check establishes whether the user who is logged on in the client system is allowed to log himself or herself on to the server system under the user ID in question.
- The authorization object **S_RFCACL** has the following authorization fields:

- **RFC_SYSID**

- System ID of the calling SAP Web AS system A (that is the client system)

- **RFC_CLIENT**

- Client of the calling SAP Web AS system A

- **RFC_USER**

- User ID of the calling user in SAP Web AS system A. It is important that full authorization (“*****”) is not entered here, or else every user in SAP Web AS system B would be able to call the function groups in the **S_RFC** object. This could be a critical problem if, for example full authorization has been defined, as is often the case. It is precisely because

of this problem that in many cases trusted RFC calls are not used. It is, therefore, recommended that this type of call only be used in exceptional cases.

- **RFC_EQUSER**

- Indicator that shows whether the RFC service user can be called only by a user with the same ID ("Y" = Yes, "N" = No).

- **RFC_TCODE**

- A calling transaction code of minor importance

- **RFC_INFO**

- Additional information from the calling SAP Web AS system that is currently inactive.

- **ACTVT**

- Activity. Currently, this field can take only the value "16" (execute).

As mentioned already, from the security point of view, trusted RFC calls are more problematic, and so its wide usage should be avoided.

- It is important that the correct form of authorizations be used for the **RFC_NAME** authorization field of the **S_RFC** object. Full authorization should be avoided at all costs in this case. However, implementation instructions often only allow for full authorization. The concept described below is a way of circumventing this problem, as it allows to specify the function groups that needs to be called. This is done using the Security Audit log, which is activated using transaction **SM19** and can be analyzed using transaction **SM20**. Unlike the system trace, (transaction **ST01**), the Security Audit Log is not particularly resource-intensive, so it can remain active over quite a long interval of time.

- The Security Audit Log has to be activated for all clients, users and events in the "RFC calls" audit class. This is done in transaction **SM19**. After approximately two months – we can assume that in this time period, the function groups that are necessary for the application will have been called at least once – the audit log can be analyzed, using transaction **SM20** . The log can then be downloaded to an Excel table and the called function groups filtered out. These can then be entered into **RC_NAME** authorization field. The scope of the authorization is thus set to exactly these function groups.

- This procedure should be followed for all documented RFC call destination, so that full authorization can be gradually eliminated.

Please Let me know if any concerns.

Thanks,
Jays
http://sapsecurity.wordpress.com/
