AUTOMATION ANYWHERE | Bot Store

SAP Remote Automation

Readme

Version 1.0

05/07/2020

Table of Contents

1. Introduction

The following documentation will give you an overview of the data architectures of the SAP Remote Automation meta-bot (SRA). It will describe the high-level concepts that you need to be familiar with in order to create task-bots successfully.

1.1 Overview

SRA provides consistent access to SAP data through remote function calls (RFC). AAI Task bots can use SRA to connect, retrieve, handle, and update the data within SAP.

SRA separates data access from data manipulation into discrete components that can be used separately or in tandem. SRA includes data structures for connecting to SAP, executing RFCs, and retrieving results. Data input and data output is placed in SRA's in memory persistence in order to be exposed to the task bots in an ad hoc manner.

The SRA meta-bot logic can be found in SapRemoteAutomation.dll and are integrated with the AAI in SapRemoteAutomation.mbot. For sample code that connects to SAP, retrieves data from it, and then exports that data to a file, see the SRA task bot example.

SRA provides functionality to task-bot developers who require an advanced yet streamlined approach to retrieving or updating data within SAP.

SRA provides the most direct method of data access from a remote position.

1.2 Common Use cases

This bot performs the following:

- Executing SAP RFCs (BAPIs).
- Integrate external applications with SAP.
- Migrate data from one SAP instance to another. Export data from SAP Table to CSV.

2. Requirements & Prerequisites

2.1 System Requirements

- Automation Anywhere Enterprise v11 or higher.
- Microsoft C++ Runtime DLLs version 10.0. Microsoft download page

2.2 Prerequisites

- Automation Anywhere Enterprise v11 or higher.
- Microsoft C++ Runtime DLLs version 10.0. Microsoft download page

2.3 Security Measures

The SAP system expects to use ports 3200 through 3299 and 3300 through 3399 for SAP gateways.

2.4 Disclaimers

Configuration of SAP environments and computers accessing them is unique to each environment. Although this document guides you through all configuration's possible, it can only describe all options and cannot be specific to your environment. Contact your SAP support staff for further guidance.

3. Getting Started

3.1 Skill Matrix

Skill	Task File	MetaBot Files
Exports a table from SAP.	Example.atmx	SAP_Metabot.mbot

Skill	MetaBot Files
Connect with SAP instance to remotely execute a BAPI.	SAP_Metabot.mbot

3.2 Installation Hierarchy

<AA Application Path> 1. My Tasks 1.1. Bot Store 1.1.1. SAP Remote Automation-SiriusIQ • Error Folder o Logs o Snapshots • Input Folder • My MetaBots o SAP_Metabot.mbot • My Tasks o Example.atmx	

3.3 Quick Start

3.3.1 Setup

If SAP is configured to connect to a gateway on System 01, then it will use driver on port 3301. If the System is 11, then port 3311 is expected. If another SAP product is not installed on the machine which will be running this meta-bot, then you must manually configure the entries in

Appendix D for port names in <Drive>:\WINDOWS\system32\drivers\etc\services file. This is simply done using the following steps.

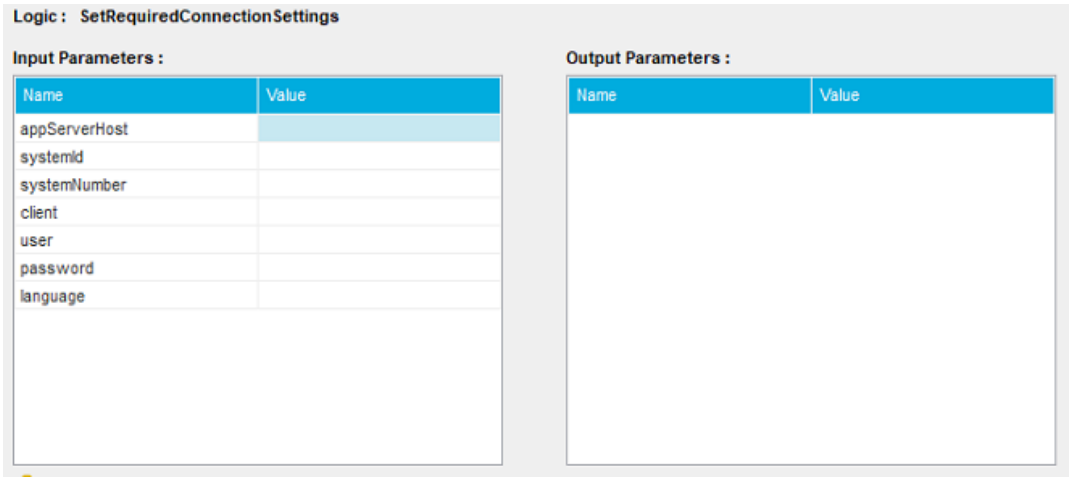
1. Navigate to the <Drive>:\WINDOWS\system32\drivers\etc\ directory.
2. Open the services file using notepad.
3. Insert the contents of Appendix D to the end of the file.
4. Save the file.

3.3.2 Configuration

SRA makes logging on to SAP incredibly easy using two meta-bot logic functions.

1. **SetRequiredConnectionSettings**

   This is a simplified function allowing you to easily provide a minimum set of connection parameters. All values are required for this function.

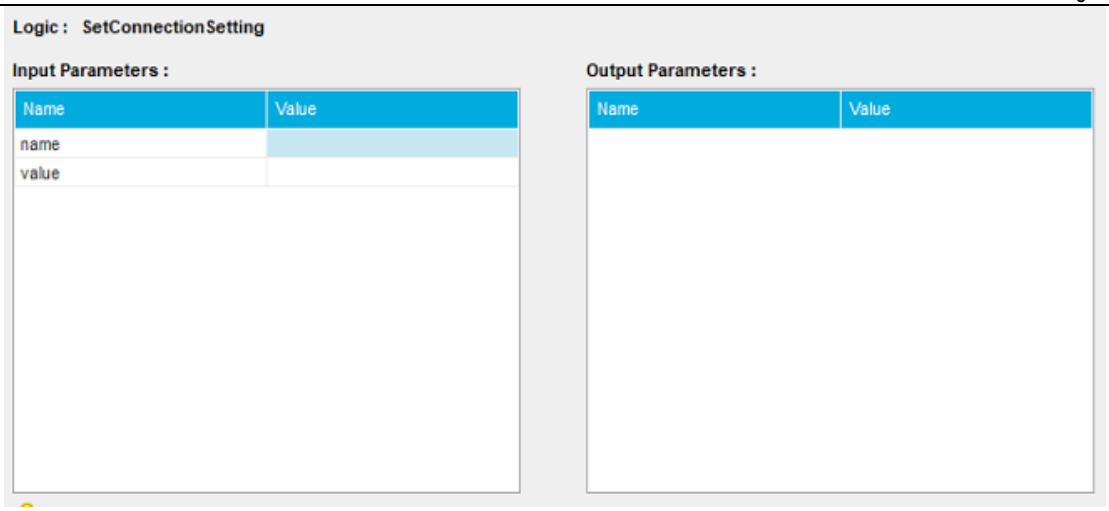


   *Although most system use the settings provided in the SetRequiredConnectionSettings function, this is not always the case and use of this function is optional. Your SAP support staff should be able to provide you with exact connection parameters.

   * Use of an SAP router is a common connection scenario that is not in the function above. In this case, it is possible to call SetRequiredConnectionSettings and then call SetConnectionSetting passing in the Name 'SAPROUTER' and value (Example: /H/255.255.255.255/H/).

2. **SetConnectionSetting**

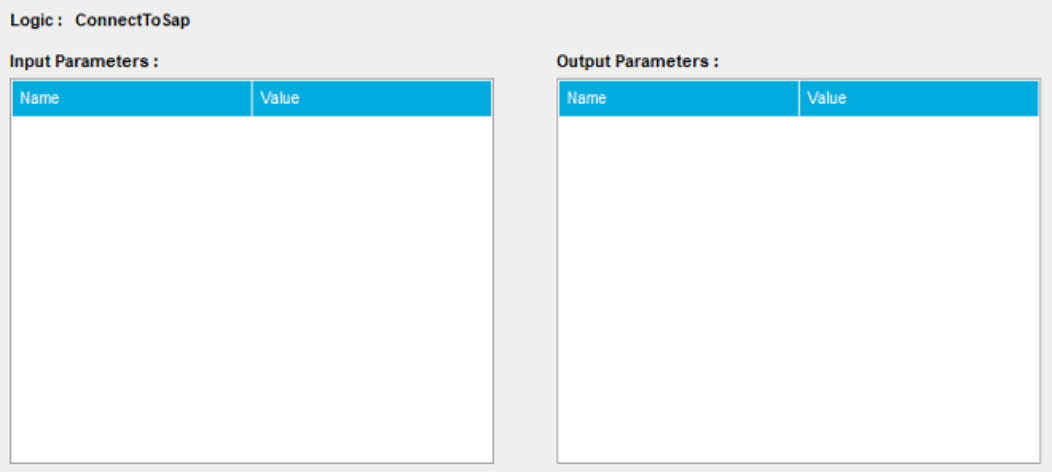
   This allows the input of a number of connection setting options. These options can be found in Appendix E of this documents. This function will be called several times, once for each connection parameter name and value combination, based on the connection requirements for the SAP instance.



INPUT VARIABLES: Refer to Appendix E for full list of parameters.				
Variable Name	Type	Mandatory	Purpose	Example Input
systemId	Text	Yes	The SAP system's three-letter system ID	S18
systemNumber	Text	Yes	The SAP system's system number (00-99).	00
client	Text	Yes	The client into which to log in (000-999).	100
user	Text	Yes	The username to be used for log in.	jjohnson
password	Text	Probably	The password to be used for log in. This should be suppled from the vault.	$Password
appServerHost	Text	Yes	The host name of the specific SAP application server, to which all connections shall be opened (URI or IP address).	255.255.255.255
language	Text	Yes	The 2-digit language abbreviation to be used.	EN

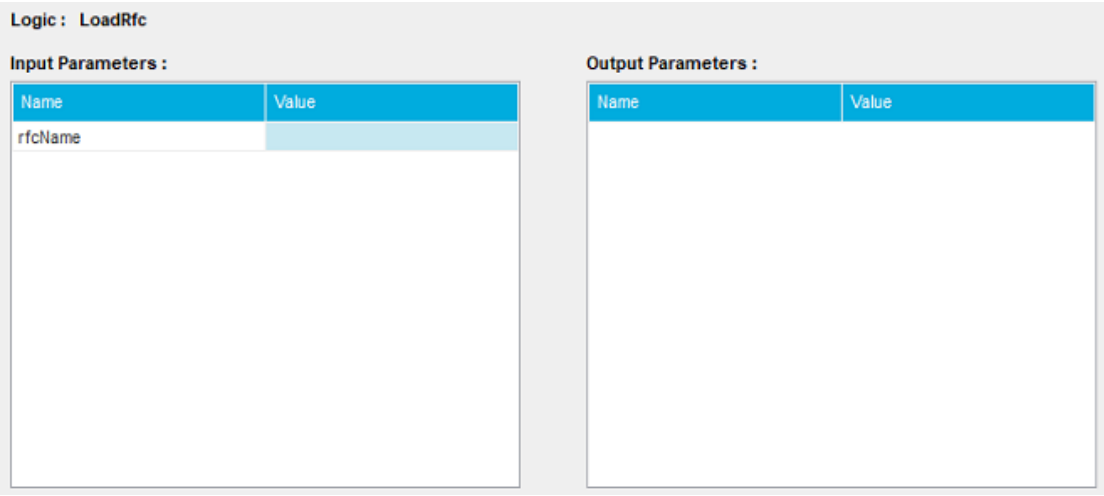
3.3.2.1 Connecting to SAP

A connection to sap is established through the SRA's 'ConnectToSap' logic function. This function does not have any parameters itself but will fail if the connection parameters have not been set or are incorrect.



3.3.2.2 Loading an RFC

To work with the Schema of an RFC, it must be loaded. A given task bot can only load a single RFC at a time. This is a per task-bit instance limitation only and other task bots can run simultaneously. The logic function 'LoadRfc' takes the name of the RFC to load. Your task bot can now begin to work directly with the schema. Note that 'LoadRfc' does not execute the RFC or manipulate SAP data.



3.3.2.3 Working with RFC Schema

An RFC contains three types of data.

- Field - simple data type such as a string value.
- Structure - complex data type containing multiple fields.
- Table - a collection of related data held in a table format.

3.3.2.3.1 Working with Fields

Fields are the simplest construct within an RFC schema. It has a name, datatype, and value. Specifying the field name is all that is required when working with fields that are direct members of the RFC schema (I.E. fields that are not part of structures or tables).

3.3.2.3.2 Working with Structures

Structures are a complex type containing multiple fields. The structure is made up of a single name and at least one field. Dot notation is used when requesting a field from a structure. [StructureName].[FieldName] (Example: Company.Name).

3.3.2.3.3 Working with Tables

Like structures, a table has a complex structure. These are known as columns, where each cell represents a value. Each row represents another instance of that complex structure. Rows are always referenced by index (an integer), but columns can be referenced by name or ordinal position (an integer).

3.3.2.3.4 Accessing Data within an RFC Schema

Before executing an RFC, certain information may need to be provided. This could be filters or data you wish to import. After executing an RFC, certain data needs to be retrieved, such as a result set. When working with an RFC, all data (for input or for output) can be accessed using the path naming convention.

[FieldName] for fields (Example: FirstName).

[StructureName].[FieldName] for structures (Example: Company.Name).

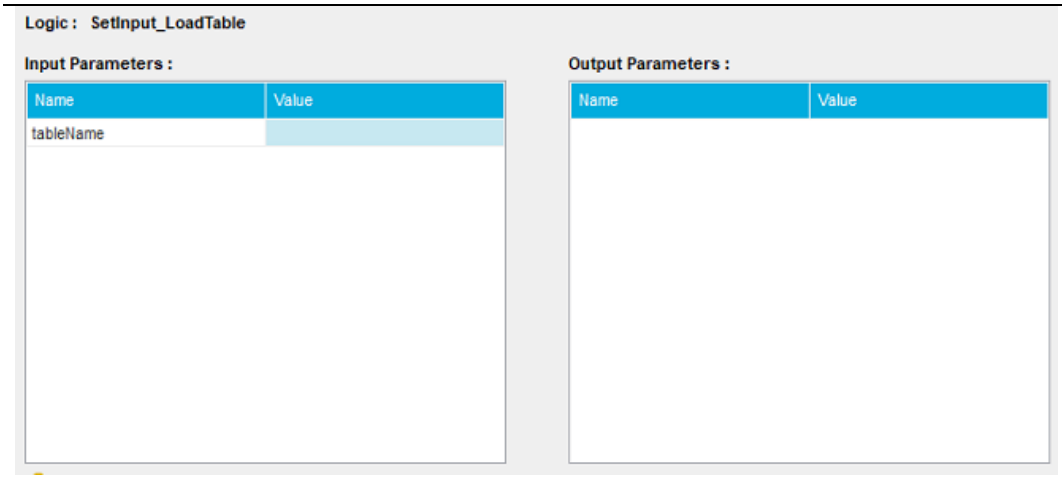
[RowIndex] and [TableName].[ColumnName] for tables (Example: 0, Invoices.Date). *Tables can also be iterated using row index and column ordinal for fast data extraction.

3.3.2.3.5 Adding Parameters / Data to an RFC Schema

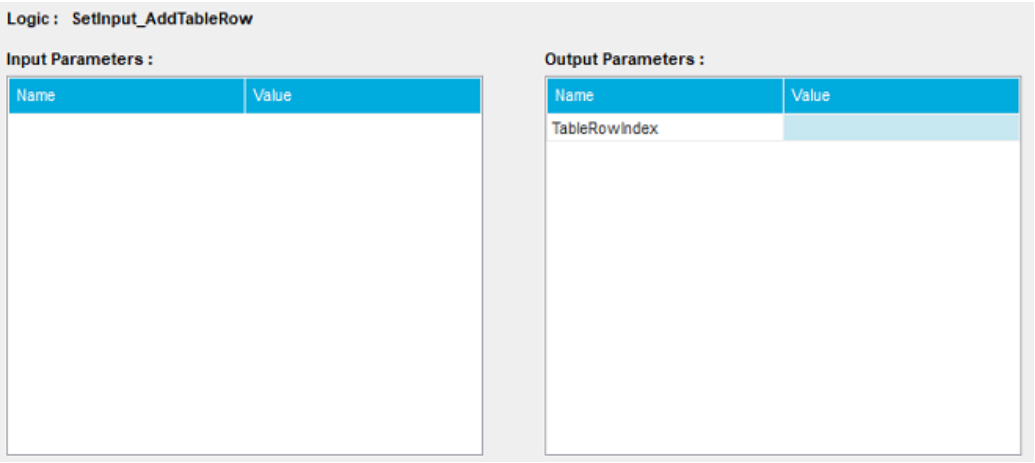
The 'SetInput' function is will set values on either a field, structure or table within the RFC schema. The 'fieldPath' input parameter uses the naming convention detailed above to supply the value. Note that this function does not allow for the table index to be referenced. Therefore, all table inputs will be to the 0 index (first item) in the table. You cannot add multiple rows through this function.

To add multiple rows to a table you will need to use the 'SetInput_LoadTable', 'SetInput_AddTableRow', and 'SetInput_SetTableColumn' functions.

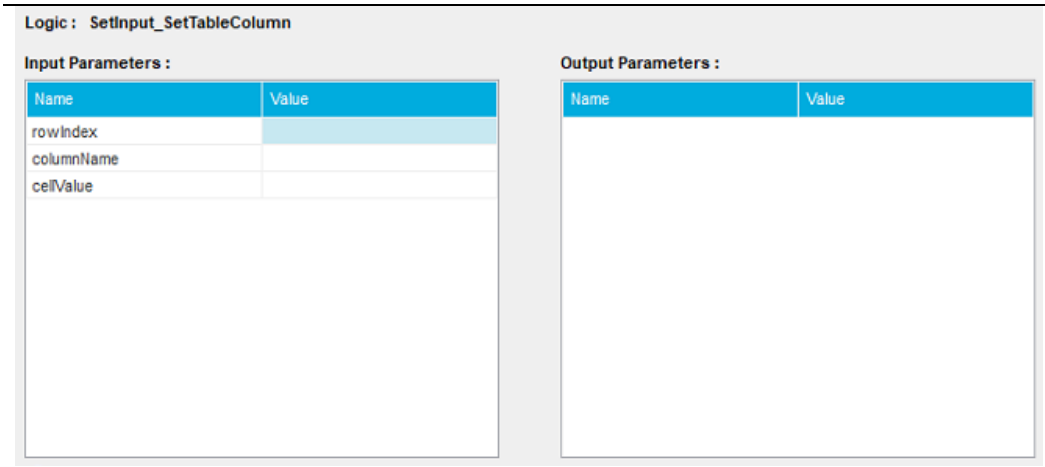
The 'SetInput_LoadTable' function is used to start the loading of a table. It lets SRA know what table it will be working with. This function only requires the table name to be provided.



The 'SetInput_AddTableRow' function is used to add a row to the table. It is impossible to work with a row until it has been created, so this function must be called. Its return value is the index of the row which was created.



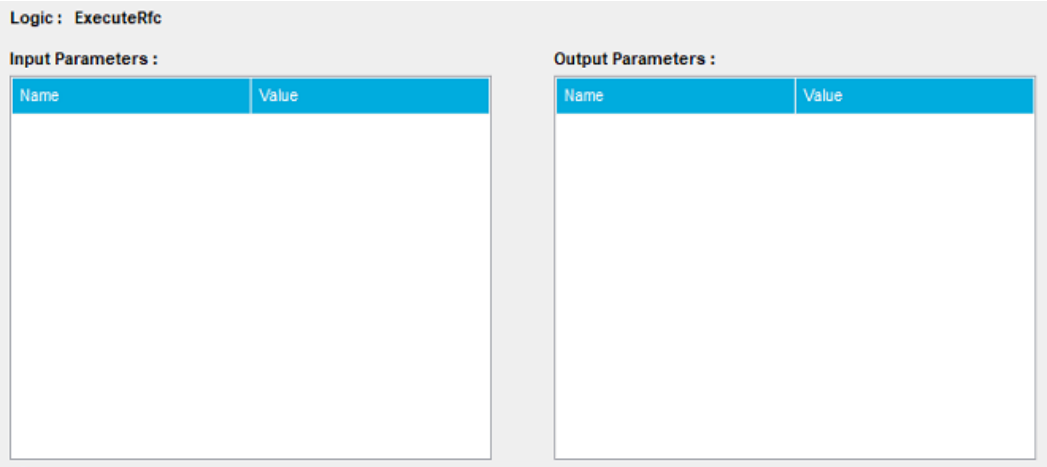
The 'SetInput_SetTableColumn' function is used to set the value of the column. This function takes three parameters: 'rowIndex'.'columnName'.'cellValue'. *Note that column name is not the path (does not use the naming convention outlined above) and should contain only the name of the column.



It is possible to load multiple tables. Each time 'SetInput_LoadTable' is called, it will change the table being loaded. The function 'SetInput_AddTableRow' is also safe to call after changing to a table that already contained rows.

3.3.2.3.6 Executing the RFC

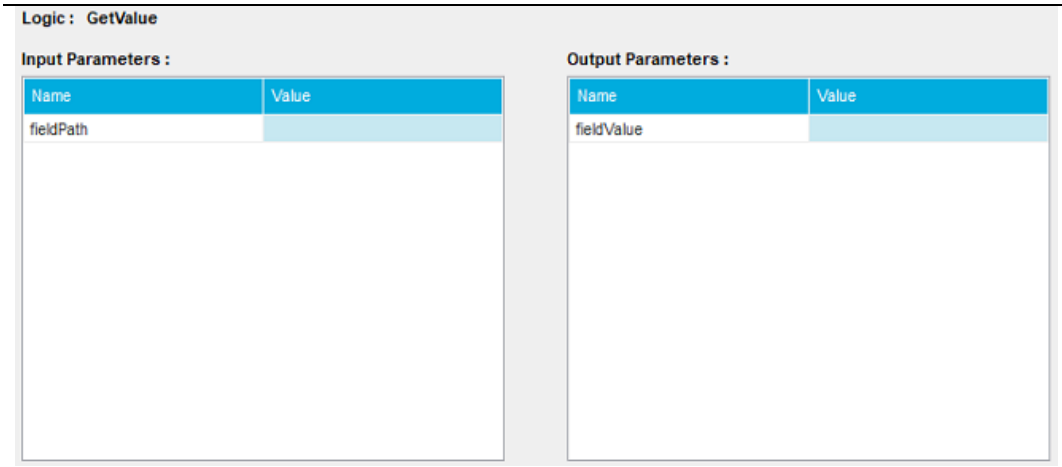
Executing the RFC is conducted using the 'ExecuteRfc' function. This function will raise an error if the RFC fails to execute.



3.3.2.3.7 Reading data from the RFC Schema

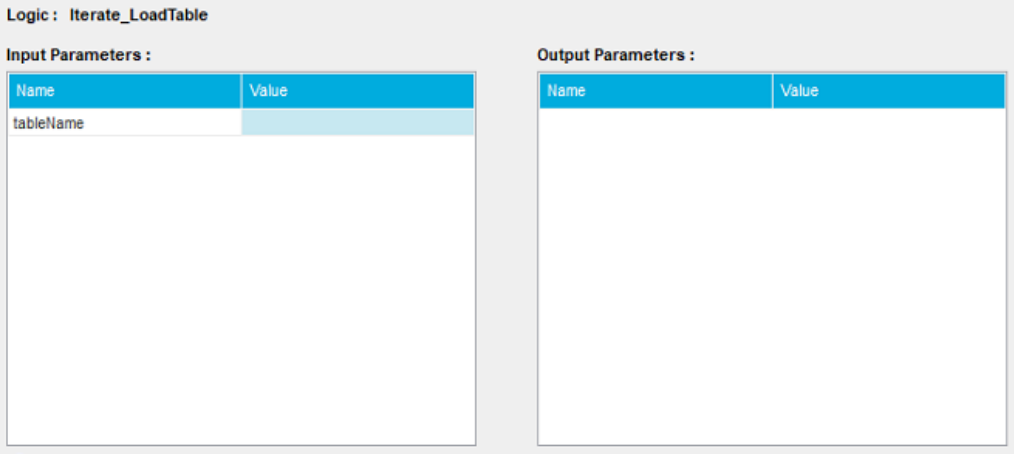
Once the RFC has executed, it is possible to read data from the schema.

The 'GetValue' function is will return values from either a field, structure or table within the RFC schema. The 'fieldPath' input parameter uses the naming convention detailed above to supply the value. Note that this function does not allow for the table index to be referenced. Therefore, all table values will be to the 0 index (first item) in the table. You cannot return multiple rows through this function.

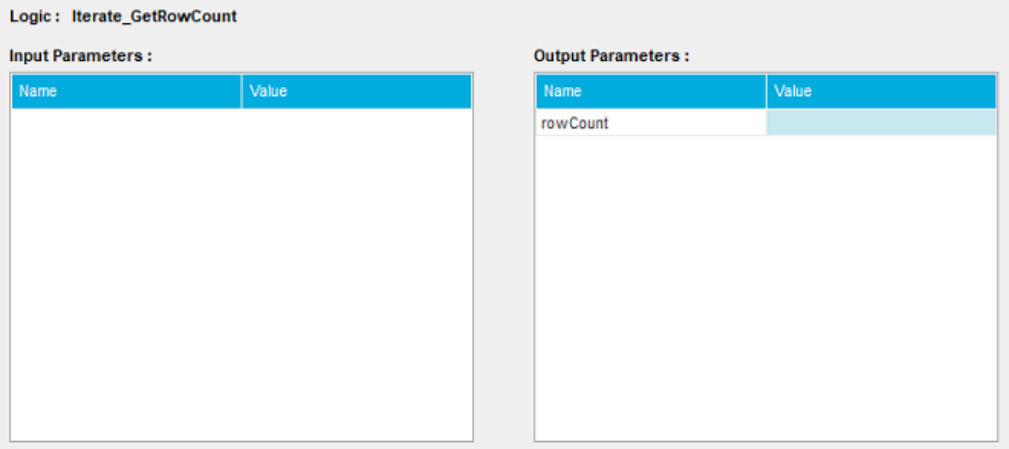


To return multiple rows from a table you will need to use the 'Iterate_LoadTable', 'Iterate_GetRowCount', 'Iterate_GetColumnCount', 'Iterate_GetColumnName', 'Iterate_GetColumnValue', and 'Iterate_GetValue' functions.

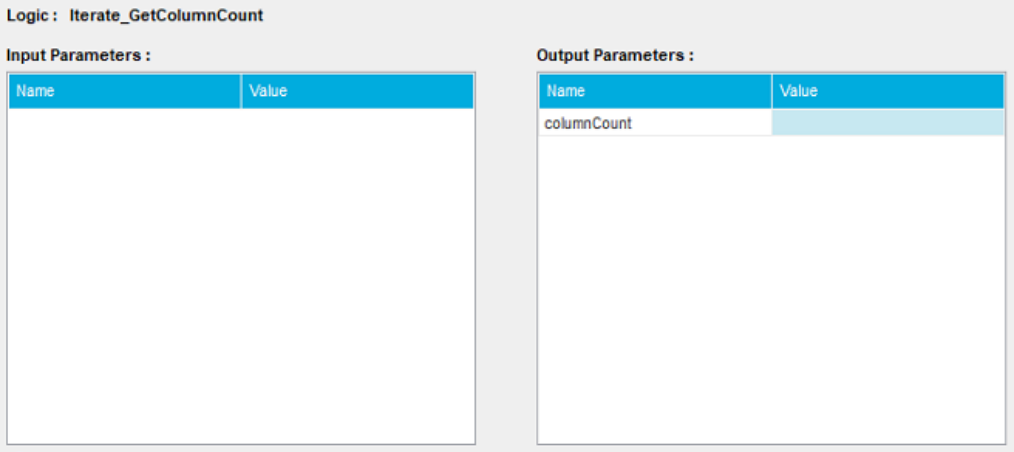
The 'Iterate_LoadTable' function is used to start the retrieval of data from a table. It lets SRA know what table it will be working with. This function only requires the table name to be provided.



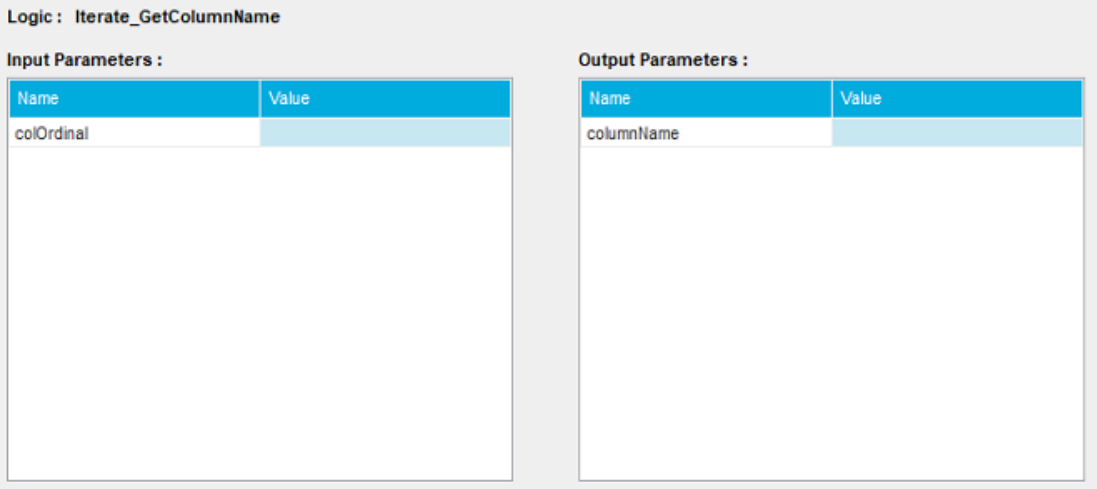
The 'Iterate_GetRowCount' returns the number of rows in the loaded table.



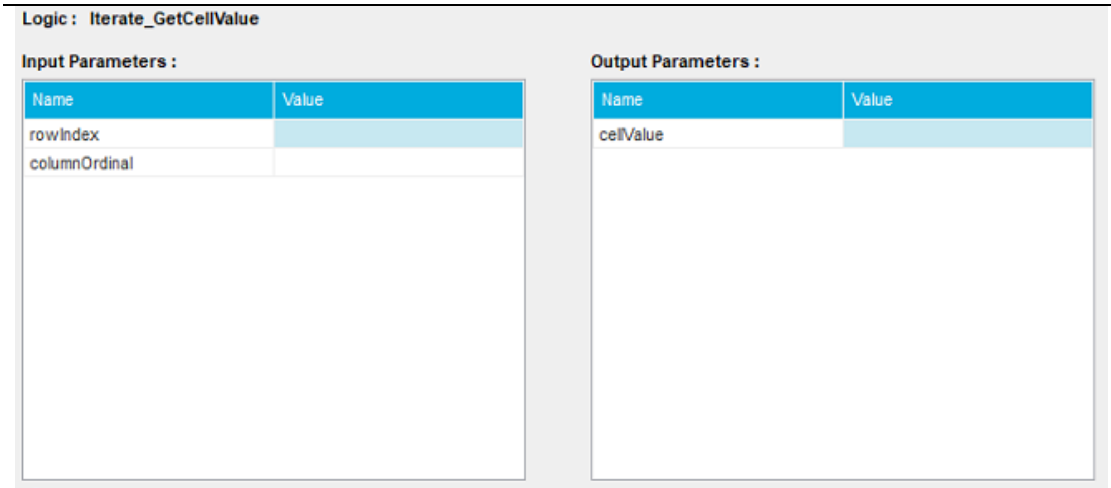
The 'Iterate_GetColumnCount' returns the number of columns in the loaded table.



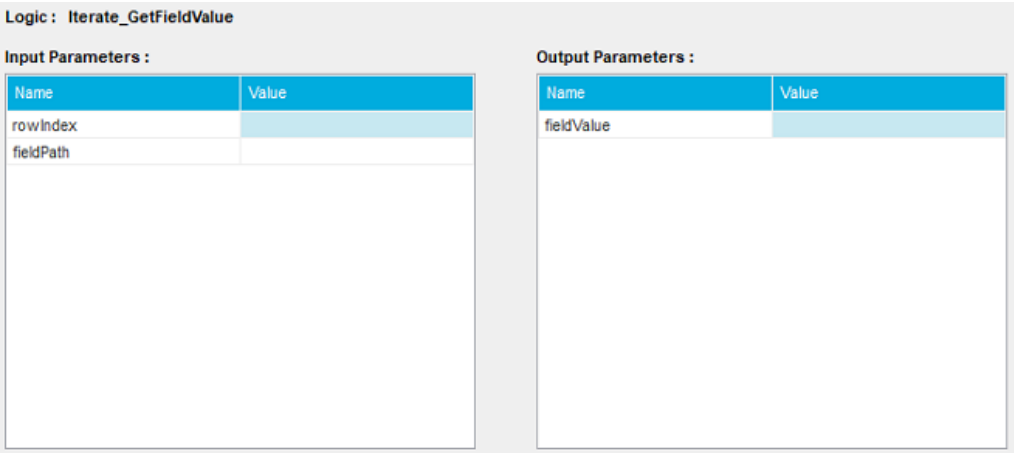
The 'Iterate_GetColumnName' returns the name of the column. This function requires the column ordinal.



The 'Iterate_GetCellValue' function is used to return the value of the column. This function takes two parameters: 'rowIndex' and 'columnOrdinal'.



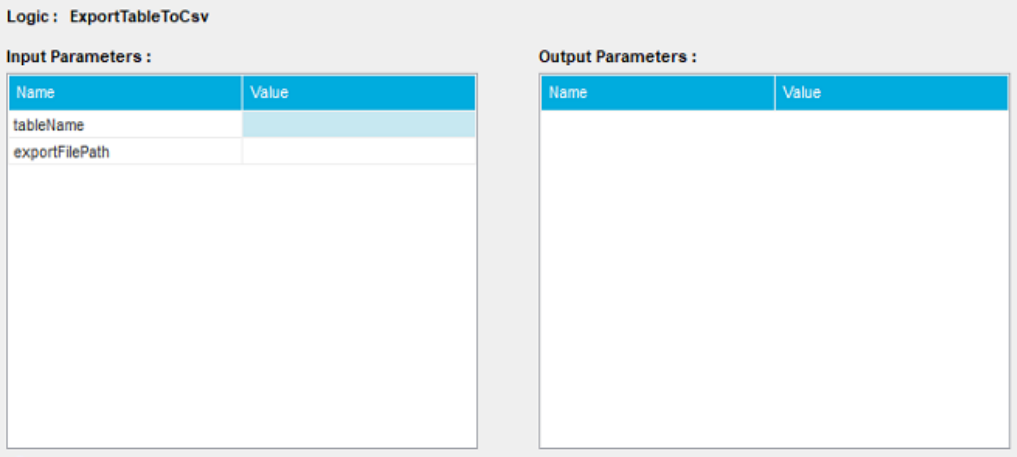
The 'Iterate_GetFieldValue' function is used to return the value of a path in regard to the row index of the table being read. If the path of the value is mapped to a field or structure, that value is returns and the row index does not come into play. If the path points to any table, it will use the row index as the row it is to retrieve. This allows for the pulling of any data within the schema, even though it is iterating a table.



It is possible to iterate multiple tables. Each time 'Iterate_LoadTable' is called, it will change the table being loaded so a new iteration can begin.

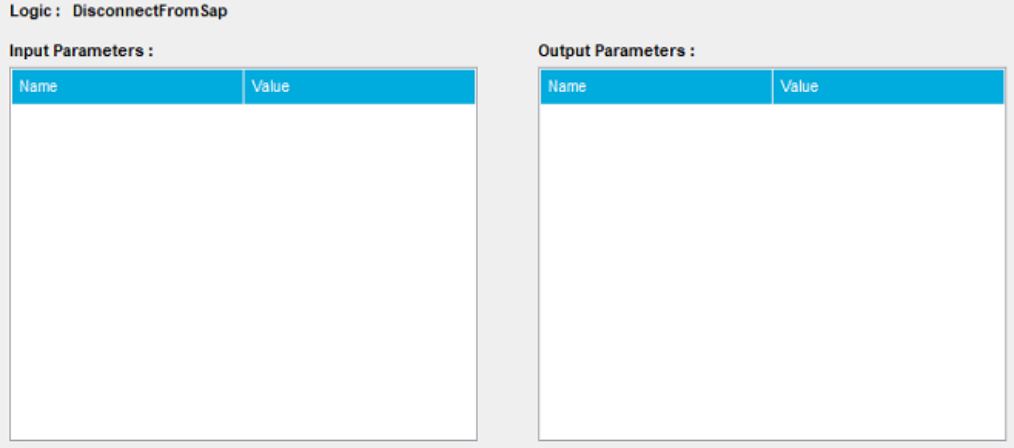
3.3.2.4 Exporting Tables

The 'ExportTableToCsv' function is used to export a single table to CSV. This is function take the name of a table and a path to save the CSV file.



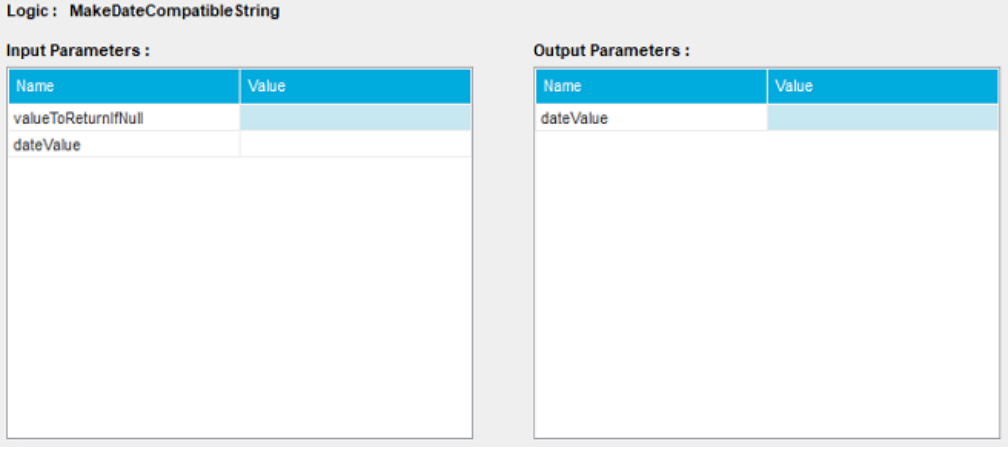
3.3.2.5 Disconnecting from SAP

The connection to sap is destroyed using the SRA's 'DisconnectFromSap' logic function.

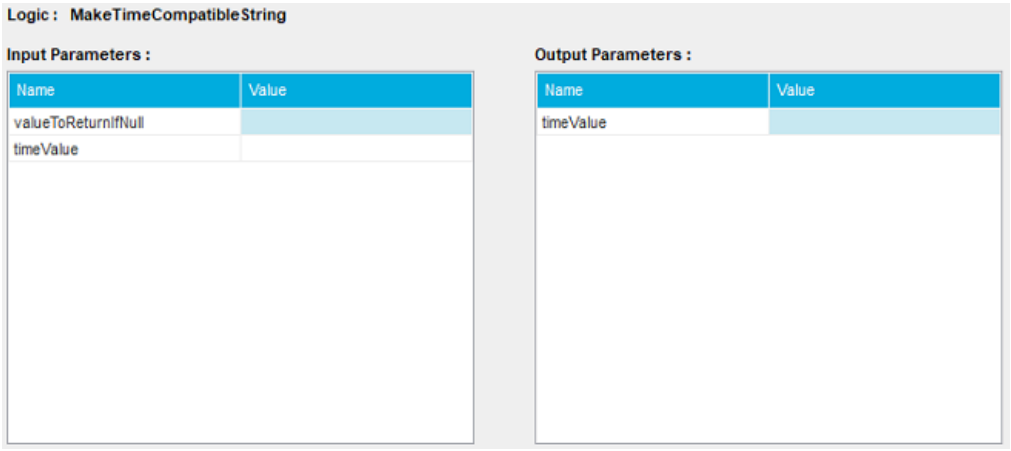


3.3.2.6 Utility Functions

The 'MakeDateCompatibleString' function will take a string representation of a date and convert it to the SAP acceptable format.



The 'MakeTimeCompatibleString' function will take a string representation of a time and convert it to the SAP acceptable format.

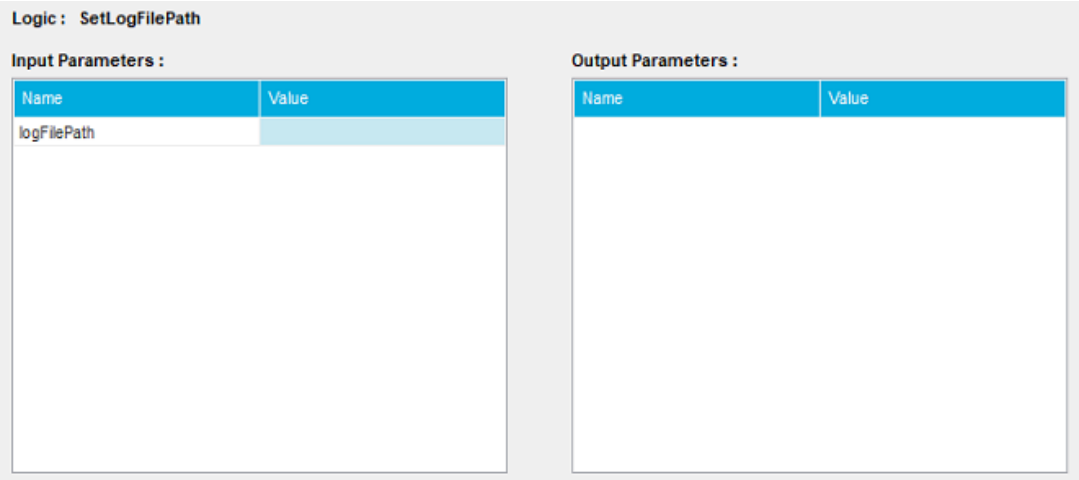


4. Reports

Data output from BAPI's can be stored as CSV files and pushed to any reporting system. BAPI creation is outside of the scope of this document. See your SAP support staff if BAPI creation is necessary.

5. Logs

SRA has the ability to log to a file for debugging purposes. To enable logging, execute the meta-bot logic 'SetLogFilePath' passing in the path to the log file where you want output saved. Use Automation Anywhere's $AAApplicationPath$ variable to identify accurate log location. I.E. $AAApplicationPath$\ErrorFolder\Logs\MyCustomTask.log.



6. Troubleshooting & Support

6.1 Support

SAP Support: Contact your SAP support team.

SAP Development: Contact your SAP support team.

SAP Remote Automation-SiriusIQ MetaBot Specific Support: support@siriusiq.com.

Task and Other metabot support: Contact Automation Anywhere support.

6.2 FAQs

Q: How do I obtain the connection settings for our SAP environment?

A: These settings are unique to your environment. If you use the SAP GUI to connect to your instance, then you might be able to obtain your connection information from there. You SAP support team should be able to provide you with an account that will allow for remote execution.

Appendix A: Record of Changes

No.	Version Number	Date of Change	Author	Notes
1	1.0	5/7/2020		

Appendix B: Acronyms

No.	Acronym	Description
1	SRA	SAP Remote Automation meta-bot.
2	RFC	Remote Function Call.
3	BAPI	Business Application Programming Interface.

Appendix C: References

No.	Topic	Reference Link

Appendix D: Port / firewall requirements

sapdp00 3200/tcp
sapdp01 3201/tcp
sapdp02 3202/tcp
sapdp03 3203/tcp
sapdp04 3204/tcp
sapdp05 3205/tcp
sapdp06 3206/tcp
sapdp07 3207/tcp
sapdp08 3208/tcp
sapdp09 3209/tcp
sapdp10 3210/tcp
sapdp11 3211/tcp
sapdp12 3212/tcp
sapdp13 3213/tcp
sapdp14 3214/tcp
sapdp15 3215/tcp
sapdp16 3216/tcp
sapdp17 3217/tcp
sapdp18 3218/tcp
sapdp19 3219/tcp
sapdp20 3220/tcp
sapdp21 3221/tcp
sapdp22 3222/tcp
sapdp23 3223/tcp
sapdp24 3224/tcp
sapdp25 3225/tcp
sapdp26 3226/tcp
sapdp27 3227/tcp
sapdp28 3228/tcp
sapdp29 3229/tcp
sapdp30 3230/tcp
sapdp31 3231/tcp
sapdp32 3232/tcp
sapdp33 3233/tcp
sapdp34 3234/tcp
sapdp35 3235/tcp
sapdp36 3236/tcp
sapdp37 3237/tcp
sapdp38 3238/tcp
sapdp39 3239/tcp
sapdp40 3240/tcp
sapdp41 3241/tcp
sapdp42 3242/tcp
sapdp43 3243/tcp
sapdp44 3244/tcp
sapdp45 3245/tcp

```
sapdp46  3246/tcp
sapdp47  3247/tcp
sapdp48  3248/tcp
sapdp49  3249/tcp
sapdp50  3250/tcp
sapdp51  3251/tcp
sapdp52  3252/tcp
sapdp53  3253/tcp
sapdp54  3254/tcp
sapdp55  3255/tcp
sapdp56  3256/tcp
sapdp57  3257/tcp
sapdp58  3258/tcp
sapdp59  3259/tcp
sapdp60  3260/tcp
sapdp61  3261/tcp
sapdp62  3262/tcp
sapdp63  3263/tcp
sapdp64  3264/tcp
sapdp65  3265/tcp
sapdp66  3266/tcp
sapdp67  3267/tcp
sapdp68  3268/tcp
sapdp69  3269/tcp
sapdp70  3270/tcp
sapdp71  3271/tcp
sapdp72  3272/tcp
sapdp73  3273/tcp
sapdp74  3274/tcp
sapdp75  3275/tcp
sapdp76  3276/tcp
sapdp77  3277/tcp
sapdp78  3278/tcp
sapdp79  3279/tcp
sapdp80  3280/tcp
sapdp81  3281/tcp
sapdp82  3282/tcp
sapdp83  3283/tcp
sapdp84  3284/tcp
sapdp85  3285/tcp
sapdp86  3286/tcp
sapdp87  3287/tcp
sapdp88  3288/tcp
sapdp89  3289/tcp
sapdp90  3290/tcp
sapdp91  3291/tcp
sapdp92  3292/tcp
sapdp93  3293/tcp
sapdp94  3294/tcp
```

sapdp95 3295/tcp
sapdp96 3296/tcp
sapdp97 3297/tcp
sapdp98 3298/tcp
sapdp99 3299/tcp
sapgw00 3300/tcp
sapgw01 3301/tcp
sapgw02 3302/tcp
sapgw03 3303/tcp
sapgw04 3304/tcp
sapgw05 3305/tcp
sapgw06 3306/tcp
sapgw07 3307/tcp
sapgw08 3308/tcp
sapgw09 3309/tcp
sapgw10 3310/tcp
sapgw11 3311/tcp
sapgw12 3312/tcp
sapgw13 3313/tcp
sapgw14 3314/tcp
sapgw15 3315/tcp
sapgw16 3316/tcp
sapgw17 3317/tcp
sapgw18 3318/tcp
sapgw19 3319/tcp
sapgw20 3320/tcp
sapgw21 3321/tcp
sapgw22 3322/tcp
sapgw23 3323/tcp
sapgw24 3324/tcp
sapgw25 3325/tcp
sapgw26 3326/tcp
sapgw27 3327/tcp
sapgw28 3328/tcp
sapgw29 3329/tcp
sapgw30 3330/tcp
sapgw31 3331/tcp
sapgw32 3332/tcp
sapgw33 3333/tcp
sapgw34 3334/tcp
sapgw35 3335/tcp
sapgw36 3336/tcp
sapgw37 3337/tcp
sapgw38 3338/tcp
sapgw39 3339/tcp
sapgw40 3340/tcp
sapgw41 3341/tcp
sapgw42 3342/tcp
sapgw43 3343/tcp

sapgw44 3344/tcp
sapgw45 3345/tcp
sapgw46 3346/tcp
sapgw47 3347/tcp
sapgw48 3348/tcp
sapgw49 3349/tcp
sapgw50 3350/tcp
sapgw51 3351/tcp
sapgw52 3352/tcp
sapgw53 3353/tcp
sapgw54 3354/tcp
sapgw55 3355/tcp
sapgw56 3356/tcp
sapgw57 3357/tcp
sapgw58 3358/tcp
sapgw59 3359/tcp
sapgw60 3360/tcp
sapgw61 3361/tcp
sapgw62 3362/tcp
sapgw63 3363/tcp
sapgw64 3364/tcp
sapgw65 3365/tcp
sapgw66 3366/tcp
sapgw67 3367/tcp
sapgw68 3368/tcp
sapgw69 3369/tcp
sapgw70 3370/tcp
sapgw71 3371/tcp
sapgw72 3372/tcp
sapgw73 3373/tcp
sapgw74 3374/tcp
sapgw75 3375/tcp
sapgw76 3376/tcp
sapgw77 3377/tcp
sapgw78 3378/tcp
sapgw79 3379/tcp
sapgw80 3380/tcp
sapgw81 3381/tcp
sapgw82 3382/tcp
sapgw83 3383/tcp
sapgw84 3384/tcp
sapgw85 3385/tcp
sapgw86 3386/tcp
sapgw87 3387/tcp
sapgw88 3388/tcp
sapgw89 3389/tcp
sapgw90 3390/tcp
sapgw91 3391/tcp
sapgw92 3392/tcp

sapgw93 3393/tcp
sapgw94 3394/tcp
sapgw95 3395/tcp
sapgw96 3396/tcp
sapgw97 3397/tcp
sapgw98 3398/tcp
sapgw99 3399/tcp

Appendix E: Connection Settings

Parameter Name	Description	Possible Values
CLIENT	The backend's client (or "Mandant") into which to log in.	000 - 999
LANG	The logon language to be used.	DE, EN, JA, …
USER	The username to be used for log in.	
PASSWD	The password to be used for log in.	
MYSAPSSO2	Instead of logging in with user and password, you can also log in with an SSO2 ticket or Assertion ticket. However, as an external application it is quite difficult to get one. Currently the only known way of obtaining an SSO/Assertion ticket is, if you are an RFC server program, and the backend sends you one. You can then use this ticket for logins to further systems that have the same user base.	Use this parameter instead of USER&PASSWD to log in with an SSO2 ticket (Single-Sign-On) or with an "Assertion" ticket (starting with backend release 7.00).
X509CERT	Another alternative way to user/password login is login via an X.509 certificate. The backend system needs to be set up accordingly and map the certificate to the corresponding user.	X.509 certificate in Base64 encoded form
SNC_SSO	When connecting via SNC, the SNC identity is typically used for authenticating the caller. If it is required to override this default behavior, then this configuration parameter can be set to 0, so that user/password information can be used for authenticating the caller, but the line is still encrypted.	0: Logon with the credentials provided on RFC level. 1: Use SNC SSO feature and logon using the SNC identity (default)

LCHECK	Determines whether the login procedure should be executed when opening an RFC connection. Not very useful, however, because you cannot do much with an RFC connection that is not logged in as a particular user.	0: do not perform the login procedure 1: perform the login procedure (default)
EXTIDTYPE	“External ID” is an old login mechanism that is no longer recommended. By default, this mechanism is disabled in the SAP system. If you want to enable it you have to set the profile parameter snc/extid_login_rfc=1. The external ID type defines what kind of data the external ID data parameter will contain.	NT: NTLM/ Windows Domain User ID: Microsoft .NET passport DN: certificate
EXTIDDATA	See ExternalIDType. Data which somehow identifies a SAP backend user. The exact format of this value depends on the value of the external ID type. For all values of ExternalIDType, the external ID data needs to be mapped to a SAP system user in the table VUSREXTID.	Some data identifying the external user, depending on the type of external ID
USE_SAPGUI	Determines whether a SAPGui session should be attached to the RFC connections of this destination. This can be useful if you want to call old BAPIs that produce Dynpro output and would otherwise dump with a DYNPRO_SEND_IN _BACKGROUND (“Screen output without connection to user”).	0: no SAPGui (default) 1: attach a visible SAPGui 2: attach a hidden SAPGui, which just receives and ignores the screen output. Note that for values other than 0 a SAPGui needs to be installed on the machine where the client program is running. This can be either a normal Windows SAPGui or a Java Gui. Additionally, the backend needs to fulfill the requirements listed in SAP note 1258724.

ABAP_DEBUG	Can be used for R/3 systems with release < 6.20, where "external breakpoints" are not yet available. The connections are opened in debug mode and the invoked function module can be stepped through in the debugger.	0: no debugging (default) 1: attach a visible SAPGui and break at the first ABAP statement of the invoked function module. Note that for debugging a SAPGui needs to be installed on the machine where the client program is running. This can be either a normal Windows SAPGui or a JavaGui. For backend releases >= 6.20 use "external breakpoints" instead (see note 668256), as this is more convenient and allows the debugger to run on any host, not only the host on which the RFC client program is running. Again the prerequisites of SAP note 1258724 need to be satisfied.
NO_BASXML	The flag that determines whether or not to use basXML format to serialize RFC calls. Thus, basXML can be suppressed even if the function module supports basXML according to its metadata.	0: Use basXML if function module and partner support basXML (default) 1: Always use classic serialization for the function module.
DELTA	The flag that determines whether delta management should be used for table parameters. With delta management, only added/deleted/modified lines are sent back by the partner.	0: Turn off delta management. 1: Delta management is turned on (default)
USE_SYMBOLIC_NAMES	Specifies whether the .NET Connector should use symbolic service names defined in /etc/services, like sapgw33, or hard-coded port numbers derived from the instance number, like 3300, when connecting to the application server.	0: use port numbers (default) 1: use server names

REPOSITORY_USER	If you do not want the same user to be used for both the “application level” function calls and the calls that lookup repository information from the backends' DDIC you can configure a separate repository user here. This allows you to separate the “application user” from the technical “DDIC user” and the respective set of RFC authorizations.	
REPOSITORY_PASSWD	Password for the above repository user.	
NAME	Each destination and server need to be given a name. The application can then access the destination/server via this name N by calling RfcDestinationManager.GetDestination (N) or RfcServerManager. GetServer(N) respectively.	

ON_CCE	"Character Conversion Error" What shall NCo do when it encounters a character that does not exist in the target codepage, a broken character, or a control character (0x00 - 0x19)?	This parameter can take three values: 0: Abort with an error message (default behavior). Note that in this case control characters (e.g. tabulator, carriage return, or linefeed characters) are not considered "illegal" and will therefore not cause an abort. 1: Copy the character in a "round-trip compatible way". The resulting output character may be "garbage" in the target codepage, but when converted back to the source codepage, it will be the original character. 2: Replace the character with a substitute symbol (usually a # character). Note that in this case the control characters are replaced as well. If you need the control characters, then you'll have to use option 0 or 1, depending on whether you want the NW RFC Lib to abort the call in case of broken characters or not.
CFIT	"Conversion Fault Indicator Token" The substitute symbol used if ON_CCE=2.	Needs to be given as hexadecimal value of a Unicode codepoint. The default is 0x0023 ("# character").
USE_SAP_CODEPAGES	Default uses Microsoft's codepage converters contained in the .NET Framework. However, for certain SAP Codepages this may lead to incorrectly translated and broken characters, for example, if the backend is running a "blended codepage" or an East-Asian codepage. In this case you can specify a list of codepages for which you want NCo to use the SAP codepage converters.	SAP codepages, e.g. 6100,8000,8340, or * for all.
SAPROUTER	If the connection needs to be made through a firewall via a SAPRouter, specify the SAPRouter parameters here.	A list of host names and service names / port numbers in the following format: /H/hostname/S/portnumber.

NO_COMPRESSION	By default the RFC protocol compresses tables when they reach a size of 8KB or more. On very rare occasions you may want to turn this off, for example if you are transporting huge integer/binary tables with "random" data, where compression would have no notable effect except for wasting CPU cycles. Or if you are trouble-shooting a certain problem and want to see the table in the trace file in humanreadable format.	0: Compress tables (default) 1: Do not compress tables
Application Server Logon		
ASHOST	The hostname of the specific SAP application server, to which all connections shall be opened.	
ASSERV	The service name (as defined in etc/services) or the port number, under which the application server's gateway process is listening for RFC requests. Note: usually this parameter can be omitted. By default, RFC uses the port number "33XY", where XY is the system number of the SAP system.	e.g. sapgw00, 3300
SYSNR	The SAP system's system number	00-99
Group Logon		
MSHOST	The hostname of the SAP system's message server (central instance).	

MSSERV	The service name (as defined in etc/services) or the port number under which the message server is listening for load-balancing requests. Note: usually this parameter can be omitted, if SystemID is specified. By default, RFC uses the service name “sapmsABC”, where ABC is the system ID of the SAP system. If specified, this default behavior is overridden.	e.g. sapmsPRD, 3600
SYSID	The SAP system’s three-letter system ID. Mandatory, if MessageServerService is not present.	
GROUP	The logon group, from which the message server shall select an application server. Please note, that though being optional, it is always a good idea to specify this parameter.	Default value is PUBLIC.