SailPoint Connector for ACF2

Version 4.0.02

Rev 1.2

Administration Guide

Copyright and Trademark Notices

Contents

Integrating SailPoint with ACF2 Source **1**

Connector Facilities 1

Connector Components in Detail 2

Installation **4**

Hardware and Software Requirements 4

Administrator Permissions 5

Pre-Installation Considerations 5

Installation Considerations 8

New Installation 10

Uninstalling the Connector 40

ACF2 Support Customization **41**

Overview of Connector for ACF2 Interceptors 41

Set Logging Options 46

Install and Customize Online Interceptor 46

Install Offline Interceptor Interface 55

Shared ACF2 Database Support 56

Support User-Defined Fields in the Logonid Record 60

Rule Set Backup 61

Supporting Mixed Case Passwords 62

Defining and Using SailPoint Connector for ACF2 Groups 62

Secured Communication **63**

TLS Secured Communication 63

Transmitted Data Encryption 67

Supporting Incoming IP Address Validation 68

Operations **72**

Starting Connector for ACF2 72

Shutting Down the Connector for ACF2 73

Starting and Stopping the Online Interceptor 73

Starting the Offline Interceptor Manually 74

Scheduling the Offline Interceptor 74
Stopping the Notification and Transaction Servers 77
Restarting the Notification and Transaction Servers 77
Viewing System Status 77
Connector for ACF2 Implementation 79
Multi-Valued Logonid Fields 79
ACF2 Group Representation 80
Implementing Groups 84
Connector and ACF2 Interaction 88
Scripts 89
Writing a Script 89
Executing a Script 91
Script Variables 92
Setting the Return Code 98
TSO Considerations 98
Script Commands 99
Maintenance 103
Formatting the Diagnostic Level Dataset 103
Displaying Local Connector for ACF2 Data 103
Setting Transmitted Data Encryption 104
Setting Stored Data Encryption 105
Formatting the Offline Interceptor Dataset 107
Recovering the Offline Interceptor After Failure 107
Initializing the Connector Queue 109
Changing the Size of the Connector Queue 109
Printing the Connector Queue 110
Renaming a Managed System 110
Filtering Interception Messages 112
Interception Acknowledgment 113
Maintain Custom Field-Related Keywords 114
Remove Custom Fields Support 114

Appendix A: Maintaining Connector for ACF2 using SMP/E **116**
Packaging of Connector for ACF2 using SMP/E 116
Connector for ACF2 Maintenance Procedures 118
Appendix B: Connector for ACF2 Configuration Parameters **121**
CTSPUSR – Connector for ACF2 Parameters 121
RSSPARM – Managed System Parameters 122
RSSAPI – Connector Entries and Scripts 133
CTSPARM – Assembler Format Parameters 133
Appendix C: Connector for ACF2 Datasets and JCL Procedures **135**
Connector for ACF2 Dataset List 135
Connector for ACF2 JCL Procedures 138
Appendix D: Copying a Connector for ACF2 Installation **140**
Installation Copy Procedure 140
Appendix E: Managed System-Specific fields **146**
Description of Table Column Titles 146
User Fields 147
Group Fields 159
Appendix F: Connector for ACF2 Batch Utility **160**
Security Requirements 160
Input Control Statements Syntax Rules 161
Environment Definition Syntax 161
Batch Provisioning and List Requests 162
Invocation JCL 168

Integrating SailPoint with ACF2 Source

Operating as a security administration Connector running on the various platforms in the enterprise, Connector processes and transfers security commands and data between managed system and SailPoint. SailPoint Connector for ACF2 includes the following features:

- Full aggregation
- Provisioning
- Monitoring of ACF2 activities to update SailPoint in real time with User and password changes

The following topics are discussed in this chapter:

Connector Facilities **1**

Connector Components in Detail **2**

Connector Facilities

The Connector facilities enable the Managed System (MS) to be monitored and managed by SailPoint. The Connector facilities include:

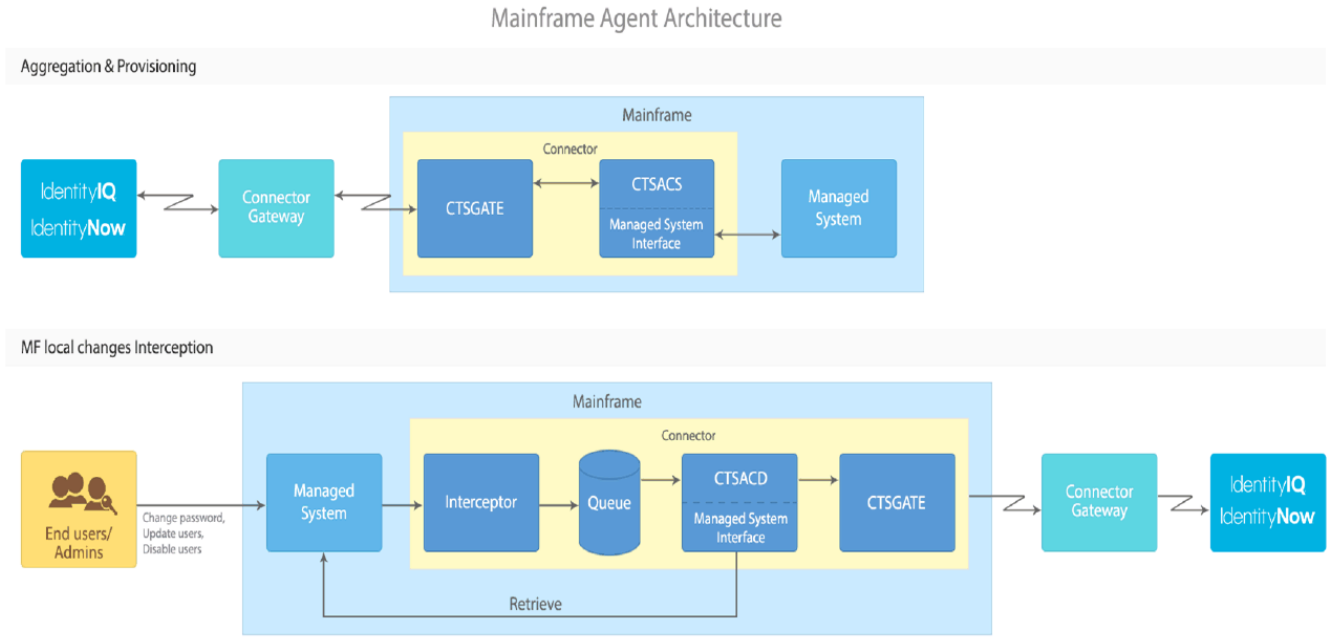
- **Managed System data aggregation to SailPoint** – The data aggregation procedure which is initiated from the SailPoint is controlled by the Connector. The Connector ensures that relevant data is aggregated from the ACF2 Managed System database to the SailPoint database. After all the data of the ACF2 Managed System has been aggregated, a consolidated picture of SailPoint data can be viewed in SailPoint.
- **Translation and execution of SailPoint commands** – Security-related commands (for example, add user, change password or phrase) which are initiated by SailPoint are handled by the Connector. The Connector translates these commands into the format and language recognized by the Managed System and executes them in the Managed System.
- **Managed System activity monitoring** – The Connector intercepts events that occur in the managed system which are initiated from within the platform environment. For example, the MS administrator adds, modifies, or deletes MS users and groups or an MS user changes his/her password or Managed System administrator changes a password for a user. When a significant event occurs, either data defining the event or an up-to-date updated entity is sent by the Connector to IdentityIQ. This functionality is accomplished using the Online Interceptor component.

> **Note**
> Online Interceptor requires IdentityIQ setting. For more information, see *SailPoint Quick Reference Guide for Gateway Connectors*.

- **Secured Communication** – Secured communication can be performed externally using AT-TLS or internally using Transmitted Data Encryption. For more information, see Secured Communication.
- **Stored Data Encryption** – All sensitive data which is stored temporarily in Connector for ACF2 (for example, sensitive security information that is written to the Connector queue file) is encrypted using a stored data encryption key.

Connector Components in Detail

The following diagram illustrates the major components of the Connector, their relationship with one another, and the flow of data between them. This diagram represents the connection between SailPoint and a single Connector installation with a single MS. In practice, multiple MS on different platforms can be administered by multiple Connector installations.



The components illustrated above work together as described in the following table.

Component	Description
Connector Gateway	Resides between SailPoint and Mainframe Connector (CTSGATE) and is responsible for the communication between these two components.

Component	Description
Connector	Enables the interception of managed system events and the translation of SailPoint commands to each specific managed system terminology. The Managed System Interface component of the Connector is a flexible API which is customized for each managed system.
CTSGATE	Mainframe side communicator gateway. Responsible for communication with Connector Gateway and CTSACS /CTSACD. It is also responsible for starting and stopping CTSACS and CTSACD.
CTSACS	Transaction Server - is responsible for SailPoint transactions handling. **Note** May be 1 to 3 transaction servers.
Managed System Interface	Responsible on the interface with ACF2 itself. It translates SailPoint transactions into ACF2 commands (provisioning transactions). It uses ACF2's API to aggregate ACF2's entities from ACF2 to provisioning module.
Managed System	ACF2
CTSACD	Notification Server - Reads events written to Queue by Interceptor, retrieve relevant entity up-to-date status from ACF2 and pass entity data to CTSGATE.
Interceptor	Responsible for intercepting Mainframe local changes done by ACF2 administrators and end-users and writes them to Queue. Two types of interceptors can be used in the Connector: • **Online Interceptor** - Detects security administration events as they occur. • **Offline Interceptor** - Detects security administration events in batch.
Connector Queue	The Connector queue is a dataset in which all security data is saved before it is sent to SailPoint via the Notification Server. If communication between Connector and SailPoint fails, Managed System events continue to be stored in the Connector queue and are sent to SailPoint when communication is re-established.

Installation

This chapter provides the required information and step-by-step instructions involved in the installation of Connector for ACF2.

> **Note**
> Before beginning the installation procedure, SailPoint recommends that you review the content of this chapter, installation steps in this chapter, and the contents of the ACF2 Support Customization chapter.

The following topics are described in this chapter:

Hardware and Software Requirements 4

Administrator Permissions 5

Pre-Installation Considerations 5

Installation Considerations 8

New Installation 10

Uninstalling the Connector 40

Hardware and Software Requirements

The following table outlines the software and hardware requirements to support the ACF2 connector.

Component	Description
Hardware Requirements	The Connector operates on any hardware configuration supported by any of the supported operating systems. • Supported systems: Connector for ACF2 can be used to manage ACF2 on z/OS release 2.4 through 2.5. Software requirements are as follows: • Job Entry Subsystem: JES2 or JES3 • TSO/E • SMP/E

Component	Description
	• TCP/IP • ISPF or any other text editor that allows submitting jobs and checking their output
Disk Type	The Connector datasets may reside on any disk that is supported by MVS. For example, **3390** IBM disk type is supported.
Disk Space	700 cylinders of a 3390 device are sufficient to store the datasets installed by the Connector installation procedure.

> **Important**
>
> Any Mainframe tool which handles x37 abends should be avoided or disabled when using the SailPoint Mainframe connector.
>
> If the tool allows it, you should add SailPoint Mainframe connector to the tool's exclude list.

Administrator Permissions

To support all the supported operations for ACF2 Connector, Administrator User ID must have **ACCOUNT+SECURITY** privileges.

Pre-Installation Considerations

This section describes the issues to be considered before beginning the installation.

Communication Parameters Coordination

For ACF2 Connector to communicate successfully with a provisioning module (SailPoint), several parameters must be coordinated between provisioning module's Application or Source Definition, Connector Gateway and ACF2 Connector.

The table below summarizes these parameters. Each row in the table describes a set of parameters in all or some of the components which must be coordinated. For more information and description of Connector Gateway parameters, see *SailPoint Integration Guide* or *SailPoint Quick Reference Guide for Gateway Connectors* depending on Connector Gateway release. For full description of IdentityIQ Application definition, see *SailPoint IdentityIQ Administration Guide*. For full description of IdentityNow Source definition, see *ACF2 Source Configuration for IdentityNow*.

Summary of Required Parameter Coordination

For Connector to communicate successfully with SailPoint, the following Connector installation/environment parameters must be coordinated with parameters specified in the Connector Gateway and in SailPoint:

Parameter Name		ACF2 Connector	Connector Gateway	IdentityIQ Application Definition	IdentityNow Source definition
RSSNAME		MSCS in RSSPARM member in PARM library		MSCS Name parameter in Connector Gateway/Connector Manager Settings	Connector name in ACF2 source
RSSTYPE		RSSP_TYPE in RSSPARM member in PARM library. Must be ACF2		MS Type selected for Application Type field should be ACF2 - Full	ACF2 source
MF_PORT		PORT parameter in ECAPARM member in PARM library	**port** parameter in SM section in **init.xml** file.		
SECURED	TRANSMITTED DATA ENCRYPTION			Encryption parameter in Connector Gateway/ Connector Manager Settings	Not supported
	TLS	AT-TLS implementation	Implementation steps have to be performed	Implementation steps have to be performed	Implementation steps have to be performed

- **RSSNAME** – In ACF2 Connector, the name is set in the **DEFPARMS %RSSNAME%** parameter in the INSTALL library during installation. After installation, this name is automatically set as the MSCS name in RSSPARM member in the PARM library. The MSCS name appears in column 1 of each parameter line (unless **ALL_RSS** is set in the line).

  This name can be up to 32 characters long. However, SailPoint recommends that you use RSSNAME configurations with eight characters or fewer. If you must configure the RSSNAME parameter with more than eight characters, special adjustments must be performed at the end of the ACF2 Connector installation procedure. For more information on the additional steps, see 11 – Adjust for Longer Managed System Names.

The same name must be specified for **%RSSNAME%** and for the MSCS Name parameter in Connector Gateway/Connector Manager Settings in Provisioning Module Application Definition (for IdentityIQ) or for the Connector name in ACF2 Source definition (for IdentityNow).

- **RSSTYPE** – During installation RSSTYPE is specified in the **DEFPARMS %RSSGTYPE%** parameter in the INSTALL library. After installation, the RSSTYPE can be found in the RSSPARM **RSS_TYPE** parameter in the PARM library. For ACF2 connector, the value must be ACF2. The MS Type in Provisioning module Application definition must be **ACF2- Full**.
- **MF_PORT** – TCP/IP port number defined for the ACF2 Connector for communication with Provisioning Module. Connector for ACF2's CTSGATE uses two consecutive TCP/IP ports to communicate with Provisioning Module. By default, the ports used are 2470 and 2471. Verify that these ports are not already in use. If they are in use, locate two other consecutive ports which are available.

  Specify the lower of these two ports when you are instructed to provide a value for parameter PORT during the Connector installation. The same port number must be specified in the port field in the SM section of the Connector Gateway **init.xml** file. for more information, see 9 – Customize Communication Settings.
- **SECURED_COMMUNICATION** – Secured communication can be implemented by using Transmitted Data Encryption or TLS. One of the following options can be selected. When TLS is selected, Transmitted Data Encryption must be set off in all components.
  - **TRANSMITTED DATA ENCRYPTION** – Communication security is gained by encrypting the transmitted data using an encryption key dataset.

    For more information, see *9.4 – Set up Secured Communication* in 9 – Customize Communication Settings.
  - **TLS** – Communication is secured using TLS. Requires implementation of steps in all components.

    In the Mainframe, TLS communication must be configured using AT-TLS. With AT-TLS, the TLS processing is performed by TCP/IP and is transparent to the application (CTSGATE). Hence no settings are required in Connector for ACF2 parameters, except for setting the Transmitted Data Encryption to **Off** as described above.

    For more information, see Secured Communication.

ACF2 Considerations

Consider how to protect temporary datasets as you configure the Connector for ACF2.

Protecting Temporary Datasets

Temporary data sets are considered protected from any access except by the job or session that created them, and hence are not required to be protected by ACF2. These files are allocated as new files, held with exclusive SYSDSN

ENQ. However in situations like system failure, a temporary data set could be left unprotected. Such a file can be accessed by any user, unless protected by ACF2.

The Connector for ACF2 handles the temporary file allocated by the EXECOUT DD statement in a special way, which causes failures due to security violation when temporary files are protected by ACF2.

The EXECOUT file is defined in the connector procedure and is created when the connector starts, under the connector User ID. But, when processing requests received from SailPoint, this file is accessed by the Connector for ACF2 under the Managed System Administrator User ID. When the temporary files are protected by ACF2, the only user that can access temporary files is the user that allocates them. So, when the connector tries to write to the file under the Managed System Administrator User ID. If this User ID does not have permission to access the file, ACF2 fails the request.

During installation, the EXECOUT DD statement allocates the file on VIO. Temporary datasets allocated to VIO are not protected by the ACF2 so no error occurs. But, if VIO is not used in the system, or if VIO is not allowed for large files, the problem occurs.

The above problem can be prevented as follows:

- Allow the files allocated by the Connector for ACF2 to be on VIO.

  Or

- Allocate the EXECOUT DD statement to a permanent file. The Connector for ACF2 utilizes source JCL for started tasks (STCJOB) for starting the procedures to maintain these permanent datasets.

To use this option, set the name of the STCJOBs library (defined in IEFJOBS DD statement in MSTJCLxx system parameter) to which Connector for ACF2 STCJOBs would be copied, as the value of the **%STCJOBS%** DEFPARMS parameter.

Installation Considerations

This section describes the issues to be considered during the installation.

Enhanced Data Integrity Considerations

If Enhanced Data Integrity function is active, ensure that all Connector for ACF2 files are set in its Exclude list.

For more information, refer to the following:

- *z/OS DFSMS Using Data Sets*
- 'IFGPSEDI (enhanced data integrity)' section in *z/OS Initialization and Tuning Reference Guide*.

Using STCJOBs

STCJOBS can be used to start the Connector for ACF2 started tasks. They are required when temporary datasets are protected (see *Protecting Temporary Datasets* in ACF2 Considerations), but can be used regardless of this protection.

To use STCJOBs, specify the name of the system library for source JCL for started tasks (STCJOB) to which the Connector for ACF2 STCJOBs would be copied. This library must be defined in the IEFJOBS DD statement in the MSTJCLxx system **parmlib** member.

The STCJOBs would be copied to this library, while renamed using the DEFPARMS **%PROCPREFS%** value as the first 3 characters of their names.

When using STCJOBs, the Connector for ACF2 started tasks procedures can be placed in one of the following libraries, according to the DEFPARMS **%PROCLIB%** parameter value:

- Copy the Connector for ACF2 started tasks procedures to the system JCL procedures library (PROCLIB). When this option is used, the name of the system JCL procedures library, must be specified in the DEFPARMS **%PROCLIB%** parameter. The started tasks procedures would be copied to this library and renamed using the **%PROCPREFS%** value as the first 3 characters of their name. When the STCJOBs are started, the system would search the standard procedure libraries for the started tasks procedures.

  Or

- Leave the Connector for ACF2 started tasks procedures in the Connector PROCLIB and use the JCLLIB JCL statement to point to this library. When this option is used, LOCALCOPY must be specified in the DEFPARMS **%PROCLIB%** parameter. The procedures are copied to the Connector PROCLIB library and renamed using the **%PROCPREFS%** value as the first 3 characters. When the STCJOBs are started, the system would search for the procedures using the JCLLIB statement in the STCJOB.

> **Note**
>
> When STCJOBs are used:
>
> - The files allocated by EXECOUT DD statements which are temporary by default, are allocated as permanent files.
> - The IGD17054I message could be displayed which is not an error message and can be ignored.
>
> In z/OS V1R13, issuance of the IGD17054I message is controlled by the value specified for the SUPPRESS_DRMSGS parameter, in the IGDSMSxx PARMLIB member. Beginning in z/OS V2R1, issuance of the IGD17054I message is controlled by the new SUPPRESS_SMSMSG parameter, also in the IGDSMSxx PARMLIB member. For more

> information, see *z/OS Initialization and Tuning Reference Guide* of the appropriate z/OS version.

New Installation

This section describes the procedures involved in creating a new installation of Connector for ACF2.

Each of these procedures is explained in further detail later in this chapter.

New Installation Procedure – Summary

The installation of Connector for ACF2 consists of the following procedures:

Procedure	Step	Job/Member Name	Description
1	**Set the Parameter Values**		
2	**Prepare Installation IMAGE from TRS file**		
	2.1		Transfer the INSTALL.TRS file Using FTP binary
	2.2		UNCOMPRESS the TRS File
	2.3	$RECEIVE	Tailor the $RECEIVE Job
	2.4	$RECEIVE	RECEIVE the Installation **IMAGE**
3	**Allocate and Load Connector INSTALL Library**		
	3.1	$LOADINS	Copy, edit and run member $LOADINS
4	**Allocate and Load Connector for ACF2 Installation Libraries**		
	4.1	LOADCTS	Tailor member LOADCTS
	4.2	LOADCTS	Submit job to allocate and load Connector for ACF2 libraries

Procedure	Step	Job/Member Name	Description
5	**Tailor Connector for ACF2 Members with Site Parameters**		
	5.1	DEFPARMS	Assign installation parameters
	5.2	CHNGEPRS	Modify Connector for ACF2 members
6	**Copy Connector for ACF2 Procedures, STCJOBs and Other Members**		
	6.1	CPYMACF2	Submit the CPYMACF2 job to perform the copy
7	**Customize Connector for ACF2 Installation Parameters**		
	7.1	CTSPARMJ	Create CTSPARM module
	7.2	RSSPARM	Assign RSSPARM parameter values
8	**Format Connector for ACF2 Datasets**		
	8.1	FORMCTS	Edit and Run Member FORMCTS
9	**Customize Communication Settings**		
	9.1		Verify TCP/IP connectivity
	9.2	ECAPARM	Connector for ACF2 Gateway Communication parameters
	9.3		Define TCP/IP DATA file
	9.4		Set up secured communication

Procedure	Step	Job/Member Name	Description
10	**Define Connector for ACF2 in ACF2**		
	10.1	CTSACF2	Define Connector for ACF2 Started Tasks in ACF2
	10.2	CTSACF2	Set Permissions to Connector Datasets
	10.3	CTSACF2	Protect the Encryption Key Datasets
	10.4	CTSACF2	Define an OMVS Segment
	10.5	CTSACF2	Grant CTSGATE with authority to use TCP/IP stack
11	**Adjust for Longer Managed System Names**		
12	**Adjusting Managed System Administrator Attributes**		
	12.1	Provide Managed System Administrator Passwords	
	12.2	Verify Managed System Administrator permissions	
13	**Add Connector for ACF2 Libraries to the MVS Authorized Libraries List**		

Procedure	Step	Job/Member Name	Description
14	**Review the Installation**		
	14.1		Verify that the Connector for ACF2 Gateway is installed properly
	14.2		Verify that the ACF2 Connector communicates successfully with SailPoint
	14.3		Verify ACF2 Connector ACF2 interface
15	**(Optional) Local Changes Migration**		
16	**Configure Automated Startup of Connector for ACF2**		
17	**Customizing ACF2 Support**		
18	**Post Installation Checks**		

1 – Set the Parameter Values

This section describes about the various tables containing all the parameters for which values must be set.

Set the **Value** column with the selected values, which would later be used to set the installation jobs and parameters.

Datasets Allocation Considerations

- The first step of the installation process creates **IMAGE** files from which the Connector files are loaded by the later steps.

  Ensure that you select different prefix + version combinations to the **IMAGE** files (**%instpref%** in Installation Upload and IMAGE Datasets Parameters) and to the Connector files (xPREFx + xVERx in Connector for ACF2 Datasets Allocation Parameters). Otherwise, the installation fails due to duplicate datasets.

- When selecting high level qualifiers for the files, ensure the product installer has ALTER authority for these files.

- Some of the parameters below are the unit name and volume serial number to be used for product datasets allocation. If the datasets must be SMS managed, leave these parameters empty (except for SPCVOL which

must be *). If the datasets must not be SMS managed, specify the values for unit and volume serial number to ensure proper handling of the file.

Installation Upload and IMAGE Datasets Parameters

Parameter	Description	Value
Upload dataset name	The name of the dataset into which the product would be transferred using FTP.	
%xmitlib%	Name of the library into which the Connector for ACF2 uploaded file would be uncompressed.	
%instpref%	Prefix selected for Connector for ACF2 installation **IMAGE** datasets that are later used to install Connector for ACF2.	
%UNIT%	• For non-SMS managed datasets specify UNIT(unitname) where unitname is the name of DASD unit where Connector for ACF2 Installation **IMAGE** datasets would be placed. • For SMS-managed datasets, specify null	
%VOLUME%	• For non-SMS managed datasets specify VOLUME(volser) where volser is the volume serial number on which Connector for ACF2 Installation **IMAGE** datasets would be placed. • For SMS-managed datasets, specify null.	

Allocate and Load Connector for ACF2 Datasets ($LOADINS and LOADCTS Dobs)

Considerations before setting Connector for ACF2 file allocation parameters:

- Each Connector for ACF2 file and library used to install and operate the product is assigned a type according to its usage. Each type has separate allocation parameters (prefix (HLQ), version, unit, volser) and is identified by a prefix assigned to its allocation parameters variables.

  File types are:

  - Installation - IL
  - Operation – OL
  - SMP/E CSI – SPC
  - SMP/E files - SPA
  - DLIBs – SPD

  Assign values to the allocation parameters in the following table according to the file types and site standards.

- Some sites use special naming conventions for Load Module libraries. Therefore, the name of the Connector for ACF2 Load library is a user-defined parameter.
- The datasets installed by the Connector for ACF2 installation procedure are described in Appendix C: Connector for ACF2 Datasets and JCL Procedures. The total DASD space they require is listed in Hardware and Software Requirements.

Connector for ACF2 Datasets Allocation Parameters

Parameter	Description	Default Value	Value
DLPREFS	Installation **IMAGE**datasets prefix. The value of this parameter should be the same value specified for **%instpref%** above.	-	
JOBNAME	Job name prefix (1 to 6 characters) to be used for jobs submitted during the Connector for ACF2 installation process.	CTLSA	

JOBCARD	Job card data to be used for all jobs submitted during the Connector for ACF2 installation procedure. Maximum length is 43 characters. The value must not contain blanks and must be enclosed in apostrophes.
	Example of the use of JOBNAME and JOBCARD parameters:
	`JOBNAME=CTSINS`
	`JOBCARD='',CTSINST,CLASS=A,MSGCLASS=X'`
	Resulting job card:
		SA,CLASS=A,
		MSGCLASS=X	

Parameter	Description	Default Value	Value
	`//CTSINS01 JOB,CTSINST,CLASS=A,MSGCLASS=X`		
STEPLIB	The Connector for ACF2 Load Module library. Note that the last qualifier of this library name must contain a maximum of four characters.	CTLSA.V400.LOAD	
ILPREFS	High level dataset name qualifier (prefix) of the Connector for ACF2 installation libraries.	CTLSA	
ILVERS	Second level dataset name qualifier (version) of the Connector for ACF2 installation libraries.	V400	
ILUNITS	Name of DASD unit where Connector for ACF2 installation libraries will be placed. • For non-SMS managed datasets, specify a generic unit (for example, 3390). • For SMS-managed datasets, it is recommended to specify a null value (that is, specify **ILUNITS=,**).	@@@@	
ILVOLS	• For non-SMS managed datasets specify the volume serial number on which Connector for ACF2 installation libraries will be placed. • For SMS-managed datasets, if a null value was specified in ILUNITS, specify a null value for this parameter as well (that is, specify **ILVOLS=,**).	####	
OLPREFS	High level dataset name qualifier (prefix) of the Connector for ACF2 operation datasets.	CTLSA	
OLVERS	Second level dataset name qualifier of the Connector for ACF2 operation datasets.	V400	
OLUNITS	Name of DASD unit where Connector for ACF2 operation datasets will be placed. • For non-SMS managed datasets, specify a generic unit (for example, 3390). • For SMS-managed datasets, it is recommended to specify a null value (that is, specify **OLUNITS=,**).	@@@@	
OLVOLS	• For non-SMS managed datasets specify the volume serial number on which Connector for ACF2 operation datasets will be placed.	####	

Parameter	Description	Default Value	Value
	• For SMS-managed datasets, if a null value was specified in OLUNITS, specify a null value for this parameter as well (that is, specify **OLVOLS=,**).		
SPCPREF	High level dataset name qualifier (prefix) of the Connector for ACF2 SMP/E CSI dataset.	CTLSA	
SPCVER	Second level dataset name qualifier of the Connector for ACF2 SMP/E CSI dataset.	V400	
SPCVOL	• For non-SMS managed datasets specify the volume serial number on which Connector for ACF2 SMP/E CSI dataset will be placed. • For SMS-managed datasets, you may specify an asterisk (that is, specify **SPCVOL=*,**).	####	
SPAPREF	High level dataset name qualifier (prefix) of the Connector for ACF2 SMP/E datasets.	CTLSA	
SPAVER	Second level dataset name qualifier of the Connector for ACF2 SMP/E datasets.	V400	
SPAUNIT	Name of DASD unit where Connector for ACF2 SMP datasets will be placed, • For non-SMS managed datasets, specify a generic unit (for example, 3390). • For SMS-managed datasets, it is recommended to specify a null value (that is, specify **SPAUNIT=,**).	@@@@	
SPAVOL	• For non-SMS managed datasets specify the volume serial number on which Connector for ACF2 SMP/E datasets will be placed. • For SMS-managed datasets, if a null value was specified in SPAUNIT, specify a null value for this parameter as well (that is, specify **SPAVOL=,**).	####	
SPDPREF	High level dataset name qualifier (prefix) of the Connector for ACF2 SMP/E distribution libraries.	CTLSA	
SPDVER	Second level dataset name qualifier of the Connector for ACF2 SMP/E distribution libraries.	V400	
SPDUNIT	Name of DASD unit where Connector for ACF2 SMP/E dis-	@@@@	

Parameter	Description	Default Value	Value
	tribution libraries will be placed. • For non-SMS managed datasets, specify a generic unit (for example, 3390). • For SMS-managed datasets, it is recommended to specify a null value (that is, specify **SPDUNIT=,**).		
SPDVOL	• For non-SMS managed datasets specify the volume serial number on which Connector for ACF2 SMP/E distribution libraries will be placed. • For SMS-managed datasets, if a null value was specified in SPDUNIT, specify a null value for this parameter as well (that is, specify **SPDVOL=,**).	####	

DEFPARMS Parameters

Parameters	Description	Default value	Value
%HOLDCLASS%	Single-character Held output class for ACF2 Connector procedures and jobs.	X	
%DUMPCLASS%	Single-character Held output class for dumps, diagnostic messages and snaps. This type of output should go to a held output-class which is cleaned frequently (if not printed) in order not to cause spool fill-out.	X	
%WORKUNIT%	Unit name for temporary datasets (for example, SYSDA, SORTWORK).	SYSALLDA (exists in all z/OS systems)	
%BRODCAST%	Name of the system BRODCAST dataset. This dataset is required in the Connector for ACF2 started task for ACF2 API execution	SYS1.BRODCAST	
%LPALIB%	Name of the library to which modules that must reside in LPA would be copied. This dataset is optional for installation of Online Interceptor exits for ACF2.	SYS1.LPALIB	
%RSSGTYPE%	Type of security product installed. Default value must not be changed.	ACF2	
%RSSNAME%	Managed System name. The name must correspond to the Managed system name defined in SailPoint. It is	MVSACF2	

Parameters	Description	Default value	Value
	recommended to use a name which is up to 8 characters long. If the name is longer than 8 characters, follow the instructions mentioned in 11 – Adjust for Longer Managed System Names. For more information, see Communication Parameters Coordination.		
%PROCPREFS%	First three characters of the Connector for ACF2 JCL procedure, and optionally STCJOB names, after they are copied to the requested library, as specified in the **%PROCLIB%** and **%STCJOBS%** parameters. This determines the name of started tasks used by Connector for ACF2. When LOCALCOPY is set for **%PROCLIB%**, the value for this parameter must not be CTS, ACF, TSS or RCF. **Important** When **%PROCLIB%** is not DONTCOPY or LOCALCOPY, ensure that there are no procedures in the PROCLIB concatenation which start with the value set for this parameter. If STCJOBs are used, verify the same for the system STCJOBs concatenation. Refer the following tables for the inter-relations between **%PROCPREFS%**, **%PROCLIB%** and **%STCJOBS%** parameters: • Allowed Combination of Values for %PROCPREFS%, %PROCLIB%, and %STCJOBS% Parameters • Disallowed Combination of Values for %PROCPREFS%, %PROCLIB%, and	CTS	

Parameters	Description	Default value	Value
	%STCJOBS% Parameters		
%PROCLIB%	Name of the System JCL Procedure library to which the Connector for ACF2 procedures will be copied. The copy will be done while renaming the procedures using the value specified for **%PROCPREFS%** as the first 3 characters. Other possible values: • **DONTCOPY** – This reserved value can be specified to prevent the Connector for ACF2 procedures from being copied to any procedures library. **Note** DONTCOPY in **%PROCLIB%** forces DONTCOPY in **%STCJOBS%**. • **LOCALCOPY** - This reserved value can be specified to copy the Connector for ACF2 procedures to the ACF2 Connector procedures library while renaming them using the **%PROCPREFS%** value as the first 3 characters. **Note** LOCALCOPY can be used when **%STCJOBS%** value is not DONTCOPY. The procedures will be used by the STCJOBs using the JCLLIB statement. Refer the following tables for the inter-relations between **%PROCPREFS%**, **%PROCLIB%** and **%STCJOBS%** parameters: • Allowed Combination of Values for	SYS2.PROCLIB	

Parameters	Description	Default value	Value
	%PROCPREFS%, %PROCLIB%, and %STCJOBS% Parameters • Disallowed Combination of Values for %PROCPREFS%, %PROCLIB%, and %STCJOBS% Parameters		
%STCJOBS%	Name of the System library for source JCL for started tasks (STCJOB) to which the Connector for ACF2 STCJOBs would be copied. This library should be defined in the IEFJOBS DD statement in the MSTJCLxx system PARMLIB member. The reserved value DONTCOPY can be specified to prevent the Connector for ACF2 STCJOBs from being copied. DONTCOPY must not be used when temporary data-sets are protected by ACF2. For more information see ACF2 Considerations and in Using STCJOBs. Refer the following tables for the inter-relations between **%PROCPREFS%**, **%PROCLIB%** and **%STCJOBS%** parameters: • Allowed Combination of Values for %PROCPREFS%, %PROCLIB%, and %STCJOBS% Parameters • Disallowed Combination of Values for %PROCPREFS%, %PROCLIB%, and %STCJOBS% Parameters	DONTCOPY	

Allowed Combination of Values for %PROCPREFS%, %PROCLIB%, and %STCJOBS% Parameters

The following table describes the allowed combination of values for **%PROCPREFS%**, **%PROCLIB%**, and **%STCJOBS%** parameters within DEFPARMS member.

%PROCPREFS%	%PROCLIB%	%STCJOBS%	Results / notes
CTS/xxx	<proclib>	DONTCOPY	Procedures are copied with specified prefix to <proclib> and STCJOBs are not copied.
CTS/xxx	<proclib>	<stcjobs library>	Procedures are copied with specified prefix to <proclib> and STCJOBs are copied with specified prefix to <stcjobs library>.
CTS/xxx	DONTCOPY	Any value	Nothing is copied. If required customer must manually copy the procedures and STCJOBs.
xxx	LOCALCOPY	<stcjobs library>	Procedures are copied with specified prefix to Connector PROCLIB and STCJOBs are copied with specified prefix to <stcjobs library>. The procedures will be used by the STCJOBs using the JCLLIB statement.

Disallowed Combination of Values for %PROCPREFS%, %PROCLIB%, and %STCJOBS% Parameters

The following table describes the combination of values that are not allowed for **%PROCPREFS%**, **%PROCLIB%**, and **%STCJOBS%** parameters within DEFPARMS member.

%PROCPREFS%	%PROCLIB%	%STCJOBS%	Notes
CTS/xxx	LOCALCOPY	DONTCOPY	Prevented with an error message by CHNGEPRS job as the procedures cannot be started.
CTS/ACF/TSS	LOCALCOPY		Prevented with an error message by CHNGEPRS job as there are already members with these names in Connector PROCLIB.

RSSPARM Parameters

Parameters	Description	Default value	Value
CTSA_ID	Unique 4-character identifier for Connector for ACF2 to use. If more than one instance of Connector for ACF2 is installed on the platform, each instance	<xxx>R where <xxx> is the value specified for **%PROCPREFS%** in DEFPARMS	

Parameters	Description	Default value	Value
	should have a unique ID.		
RSS_TYPE	Managed System type. Specify: ACF2	ACF2	ACF2
RSS_WORK_DIR	The prefix used to dynamically allocate working datasets	By default, the prefix used consists of the following: `<prefix>.<version>.<RSS_NAME>` where: • `<prefix>` is the value specified for OLPREFS • `<version>` is the value specified for OLVERS • RSS_NAME is the value specified for **%RSSNAME%** in DEFPARMS	

Communication Parameters (CTSPUSR and ECAPARM Parameter Members)

Refer to the Communication Parameters (CTSPUSR and ECAPARM Parameter Members) before setting values for the parameters mentioned in the table below.

Parameter	In Member	Description	Default Value	Value
ENCR_EXT_ACT	CTSPUSR	Select whether Transmitted Data Encryption is required. Possible values are: • **Y** – Enables Transmitted Data Encryption. Default. • **N** – Disables Transmitted Data Encryption. If TLS would be used for secured communication, the value has to be set to **N**. For more information,	Y	

Parameter	In Member	Description	Default Value	Value
		see Secured Communication.		
PORT	ECAPARM	Lower of the two consecutive port numbers to be used for TCP/IP communication.	2470	
NUMSRV	ECAPARM	The number of Transaction Servers (CSs) defined for Connector for ACF2. This number determines the number of SailPoint requests that can be processed concurrently. Each Transaction Server handles a single request at a time. The maximum number of CS's that can be specified is 3. **Note** The number of Notification Servers (CDs) is one. This number cannot be changed.	2	
IPLIST	ECAPARM	(*Optional*) Used to validate the incoming IP addresses by CTSGATE. For full description and syntax, see Configuring Incoming IP Address Validation.	-	

2 – Prepare Installation IMAGE from TRS File

Use this procedure to prepare an installation **IMAGE** from TRS file.

2.1 – Transfer the INSTALL.TRS file using FTP Binary

1. Using FTP, upload file INSTALL.TRS from the TRS file to the target system, using BINARY format.
    - The name of the uploaded dataset must be the name set for **upload dataset name** in the Installation Upload and IMAGE Datasets Parameters table.
    - The uploaded dataset should be pre-allocated with LRECL=1024, BLKSIZE=6144, RECFM=FB
    - The dataset size is approximately 45 cylinders on a 3390 device.
2. At the command prompt, specify the following commands to upload the dataset where `<mvssystem>` is the DNS name of your host system:

```
FTP <mvssystem>
```

3. Specify the user ID and password or phrase; specify the following commands:

```
bin
put <drive>:\\Install\install.trs
'<upload_dataset_name>'
```

2.2 – UNCOMPRESS the TRS File

> **Note**
> Wait for the previous installation job to finish before continuing with this procedure.

1. Tailor the following TRSMAIN job. Replace upload_dataset_name and **%xmitlib%** with the values set in the Installation Upload and IMAGE Datasets Parameters table.

```
//UNTERSE  JOB ,'UNTERSE',CLASS=A,MSGCLASS=X
//*
//UNTERSE  EXEC PGM=TRSMAIN,PARM=UNPACK
//SYSPRINT DD   SYSOUT=*
//INFILE   DD   DISP=SHR,DSN=<upload_dataset_name>     <== Customize
//OUTFILE  DD   DISP=(NEW,CATLG),UNIT=SYSALLDA,
//       DSN=%xmitlib%,                                <== Customize
//       SPACE=(CYL,(120,10,10),RLSE)
```

2. Run the job.

   The job step should end with a condition code of `0`.

2.3 – Tailor the $RECEIVE Job

> **Note**
> Wait for the previous installation job to finish before continuing with this procedure.

1. Edit member $RECEIVE in library **%xmitlib%**. The job performs RECEIVE operations to convert the uncompressed files from XMIT format to the installation **IMAGE** format.
2. Replace **%xmitlib%**, **%instpref%**, **%UNIT%** and **%VOLSER%** with the values set in the Installation Upload and IMAGE Datasets Parameters table.

> **Note**
> Verify that your SMS does not impose attributes on the installation files by SMS DATACLAS or by pre-allocation.

2.4 – RECEIVE the Installation IMAGE

1. Run the $RECEIVE job.

   All job steps should complete with a condition code of `0`.
2. Check the job output and verify that all RECEIVE and COPY instructions ended successfully and all members were received and copied.

3 – Allocate and Load Connector INSTALL Library

Create the INSTALL library and load it from the **IMAGE** INSTALL library.

> **Note**
> Wait for the previous installation job to finish before continuing with this procedure.

3.1 – Copy, Edit, and Run Member $LOADINS

1. Copy member $LOADINS from library **%xmitlib%** to a new member in the **%xmitlib%** library. The job creates the INSTALL library and loads it from the **IMAGE** INSTALL library.
2. Edit the member. Replace **%instpref%** with the value set in the Installation Upload and IMAGE Datasets Parameters table.
3. Set the values for ILPREFS, ILVERS, ILUNITS and ILVOLS procedure parameters with the values set for these parameters in the Connector for ACF2 Datasets Allocation Parameters table.
4. Run the job.

The job step should end with a condition code of `0`.

5. Check the job output and verify that the INSTALL library members were copied successfully.

4 – Allocate and Load Connector for ACF2 Installation Libraries

Use the values assigned in the Connector for ACF2 Datasets Allocation Parameters table for setting the required values in this step.

> **Note**
> Wait for the previous installation job to finish before continuing with this procedure.

4.1 – Tailor Member LOADCTS

1. Edit LOADCTS member in INSTALL library.
2. This member contains JCL for the following:
   - Allocating Connector for ACF2 libraries.
   - Loading Connector for ACF2 libraries.
   - Performing JCL adaptations of a few installation jobs.
3. Tailor the job card.

   Specify values for all the procedure parameters, using the values set in the Connector for ACF2 Datasets Allocation Parameters table.

4.2 – Submit Job to Allocate and Load Connector for ACF2 Libraries

1. Check the JCL in member LOADCTS
2. Submit the job.

   All the job steps must end with a condition code of `0`.

> **Note**
> The following error message generated by this job is not an error and can be disregarded:
>
> ```
> "CTS914E - Modifying of cards ended"
> ```

Datasets which are allocated by the job are listed in Appendix C: Connector for ACF2 Datasets and JCL Procedures.

5 – Tailor Connector for ACF2 Members with Site Parameters

5.1 – Assign Installation Parameters

> **Note**
> Wait for the previous installation job to finish before continuing with this procedure.

1. Edit member DEFPARMS in the Connector INSTALL library.

   > **Note**
   > The member contains keywords which must be assigned the required installation values. (Keywords are parameters which start and end with percent signs; for example, **%HOLDCLASS%**.)

2. Assign values using the values set in DEFPARMS Parameters.

3. Save the member (if it was modified).

> **Note**
> Throughout this book, the samples are based on the assumption that you have defined your Started Task procedures using the default **CTS** prefix. If you are using a prefix other than **CTS**, adapt the started task name in the samples to match the prefix specified in the parameter %PROCPREFS%.

5.2 – Modify Connector for ACF2 Members

Member CHNGEPRS in the Connector INSTALL library contains JCL cards to modify Connector for ACF2 members so that they conform to the installation naming conventions.

The JCL of this job should have already been set for submission by an earlier installation procedure.

> **Note**
> The job modifies members in the Connector for ACF2 libraries according to values set in member DEFPARMS in step 5.1 and values set in the previous installation procedures. Any errors in the parameter definitions in DEFPARMS may result in a need for restarting the installation procedure from the beginning. Therefore, prior to submitting the job, verify that the parameters defined in member DEFPARMS are correct.

1. Check the JCL.

2. Submit the job.

The job updates members in various Connector for ACF2 libraries.

3. Save the member (if it was modified).

4. Scan the output of the job for information and error messages issued by the job.

   The job step must end with a condition code of `0`.

6 – Copy Connector for ACF2 procedures, STCJOBs, and Other Members

> **Note**
> Wait for the previous installation job to finish before continuing with this procedure.

This step copies members to the appropriate libraries:

- Procedures are copied to the requested procedures library (when **%PROCLIB%** is not DONTCOPY)
- When **%STCJOBS%** is not DONTCOPY:
  - STCJOBs are copied to the system STCJOBs library.
  - INCLUDE members are copied to the requested procedures library.
- Parameter members are copied to the Connector PARM library.

6.1 – Submit the CPYMACF2 Job to Perform the Copy

Member CPYMACF2 in the Connector INSTALL library copies the required members to the appropriate libraries. When copied, the procedures and, if required, the STCJOBs, are renamed using the value assigned to **%PROCPREFS%** as first 3 characters. The copy is done without replace, therefore, existing procedures or STCJOBs with the same names will not be overwritten.

> **Note**
> If you specified the value **DONTCOPY** in **%PROCLIB%** parameter (in DEFPARMS member), job CPYMACF2 does not copy the procedures and STCJOBs, if requested, to your system PROCLIB and STCJOBS libraries. Instead you must copy them manually from the Connector PROCLIB library and set the first three characters to match the value specified in **%PROCPREFS%** parameter in DEFPARMS member.

1. When ready, submit the job.

2. When the job ends, check the whole `sysout`. In each copy step, check IEBCOPY utility messages and verify that all members were copied successfully.

   Expected condition codes:

- When **%PROCLIB%** is DONTCOPY, only steps CHECCPR and COPYPARM are executed and both must end with a condition code of `0`.
- When **%PROCLIB%** is not DONTCOPY:
  - When **%STCJOBS%** is DONTCOPY, steps CHECCPR, COPYPROC, and COPYPARM are executed. Step CHECCPR must end with a condition code of `4`. The other steps must end with a condition code of `0`.
  - When **%STCJOBS%** is not DONTCOPY, steps CHECCPR, COPYPROC, COPYSTCJ, COPYSTCI, and COPYPARM are executed. Step CHECCPR must end with a condition code of `4`. All other steps must end with a condition code of `0`.

See Appendix C: Connector for ACF2 Datasets and JCL Procedures for a list of the JCL procedures and STCJOBs that are copied by this job.

> **Note**
>
> If JES3 is active in your environment, update all SYSOUT DD cards in all the procedures, and optionally STCJOBs, copied by this job. The update is to drop the whole DCB parameter from ALL SYSOUT DD cards in ALL STCs.
>
> For example, instead of:
>
> ```
> //STDMSG DD SYSOUT=&OUT,DCB=(RECFM=FA,LRECL=133,BUFNO=1)
> ```
>
> You should now have:
>
> ```
> //STDMSG DD SYSOUT=&OUT
> ```

7 – Customize Connector for ACF2 Installation Parameters

> **Note**
> Wait for the previous installation job to finish before continuing with this procedure.

7.1 – Create CTSPARM Module

Member CTSPARMJ in the Connector INSTALL library should already be set for submission by earlier installation procedures.

1. Check the JCL.
2. Submit the job.

3. The job creates load module CTSPARM in the Connector LOAD library.

   All job steps must end with a condition code of `0`.

> **Caution**
>
> If the job returns with a condition code of `12`, you may need to make the following change in the `DCB` of the `SYSPUNCH DD` statement. The issue may have been caused by a change in the Binder which occurred during an update. This file is used as the input for the Binder. Under certain conditions, this may fail. This process converts the file to a sequential file, which is handled differently, and it allows the job to process correctly under the conditions which caused the `12` return code.
>
> 1. Locate the following script:
>
> ```
> //SYSPUNCH DD DSN=&&OBJECT,UNIT=&UNIT,SPACE=(80,(200,50)),
> // DCB=(DSORG=PO,RECFM=FB,LRECL=80,BLKSIZE=800),
> // DISP=(,PASS)
> ```
>
> 2. Update it to the following:
>
> ```
> //SYSPUNCH DD  DSN=&&OBJECT,UNIT=&UNIT,SPACE=(80,(200,50)),
> //DCB=(RECFM=FB,LRECL=80,BLKSIZE=800),
> //DISP=(,PASS)
> ```

7.2 – Assign RSSPARM Parameter Values

1. Edit member RSSPARM in the Connector PARM library.
2. Assign a value to the parameters using the values set in the RSSPARM Parameters table.

8 – Format Connector for ACF2 Datasets

> **Note**
> Wait for the previous installation job to finish before continuing with this procedure.

8.1 – Edit and Run Member FORMCTS

Member FORMCTS in the Connector INSTALL library contains a job which allocates and formats the Connector for ACF2 operation datasets described in the table below. The JCL should already be set for submission by earlier installation procedures.

Consider the following parameters in the Connector for ACF2 Operation Datasets table:

- `<prefix>` – Value set for the OLPREFS parameter in the Connector for ACF2 Datasets Allocation Parameters table.
- `<version>` – Value set for the OLPREFS parameter in the Connector for ACF2 Datasets Allocation Parameters table.

Connector for ACF2 Operation Datasets

`<prefix>.<version>.QUEUE`	`<prefix>.<version>.CAREGRP`
`<prefix>.<version>.DIAGLVL`	`<prefix>.<version>.CARECNN`
`<prefix>.<version>.RSSOFLI`	`<prefix>.<version>.CAREOE`
`<prefix>.<version>.ENCRINT`	`<prefix>.<version>.CAREUSR`
`<prefix>.<version>.ENCREXT`	`<prefix>.<version>.ACF2UDB`
`<prefix>.<version>.RSSKWDS`	`<prefix>.<version>.ACF2GDB`
`<prefix>.<version>.USER.CLIST`	

1. Check the JCL and submit the job.
2. All job steps must end with a condition code of `0` except:
    - CHECKRCF – This step ends with a condition code of 4.
    - CHECKTSS – This step ends with a condition code of 4.
    - ALLOCRCF – This step is not executed.
    - ALLOCTSS – This step is not executed.
    - ALLOCAIT – This step is not executed.

> **Note**
> Message IEC031I, indicating system D37 failure, appears during execution of the JCL step FORMQUE. This message can be ignored since it is expected behavior and does not present a problem.

9 – Customize Communication Settings

Communication customization should be performed at this time to enable the Connector for ACF2 to communicate with SailPoint through the Connector Gateway (CG).

9.1 – Verify TCP/IP Connectivity

The Connector for ACF2 communicates with the Connector Gateway using the TCP/IP protocol.

A functional TCP/IP connection between Connector for ACF2 and the Connector Gateway is required. Any network topology configuration that supports TCP/IP (hardware and software) can be used, as long as TCP/IP connections can be established between Connector for ACF2 and the Connector Gateway. Connectivity should be verified before you start the Connector Gateway (for example, use the ping command, Telnet commands or other TCP/IP applications).

9.2 – Specify Connector for ACF2 Gateway Communication parameters

Member **ECAPARM** in the Connector PARM library is used to define Connector for ACF2 Gateway communication parameters.

Edit member ECAPARM in the Connector PARM library and set up the parameters using the values set in the Communication Parameters (CTSPUSR and ECAPARM Parameter Members) table.

9.3 – Define the TCP/IP DATA file

> **Note**
> There is likely no need to update TCP/IP data file to enable CTSGATE to communicate with SailPoint. You may skip this chapter and return only if you encounter an issue in establishing a communication between CTSGATE and SailPoint.

z/OS TCP/IP regards the CTSGATE started task as a client application requiring a client profile dataset. This profile dataset is referred to in MVS documentation as **hlq.TCPIP.DATA**. (**hlq** is the high-level qualifier for the dataset). This dataset is the main resolver configuration dataset as set up in the local TCP by the MVS/TCP systems programmer. The TCP/IP profile dataset contains information such as the host name, domain origin and the TCPIPJOBNAME parameter. This information identifies the TCP/IP stack to use. (For more information regarding this dataset, see the IBM document *z/OS Communications Server IP Configuration Guide*.

When attempting to locate the TCP/IP profile dataset, MVS searches using the following sequence of names:

1. `<jobname>.TCPIP.DATA` (for batch jobs and started tasks)
2. `SYS1.TCPPARMS(TCPDATA)`
3. `TCPIP.TCPIP.DATA`

When located, the dataset is dynamically allocated.

The default value assigned for the high-level qualifier for the TCP/IP profile dataset during TCP/IP setup is **TCPIP**.

If the high-level qualifier for this dataset in your system has been assigned a different value or if this dataset has not been assigned one of the standard names listed above, the dataset name must be specified in parameter TCPDATA in the CTSGATE started task. This parameter is referred to by the //SYSTCPD DD statement. This issue should be coordinated with the MVS/TCP systems programmer in your organization.

> **Note**
>
> If the high-level qualifier for TCPIP.DATA at your site is TCP01, Modify the TCPDATA parameter in the Connector for ACF2 Gateway JCL procedure (CTSGATE):
>
> ```
> // TCPDATA=TCP01.TCPIP.DATA,
> ```
>
> If the high-level qualifier of this dataset is TCPIP (the default), this Parameter must be left with its default value (NULLFILE).

9.4 – Set up Secured Communication

Secured communication can be implemented using TLS secured communication or Transmitted Data Encryption.

For more information, see Communication Parameters Coordination for description of each option before selecting the secured communication method.

1. Install the selected secured communication method using the steps described in Secured Communication.

10 – Define Connector for ACF2 in ACF2

Sample ACF2 definitions for the following can be found in member CTSACF2 in the INSTALL library. Read carefully the notes in member CTSACF2 and tailor according to site standards before submitting the job or using the commands from this member. After submitting the job or executing the commands, check the whole output and verify that all the commands were processed successfully.

> **Note**
> It is assumed that the user who installed Connector for ACF2 has full authority to all Connector for ACF2 files assigned to them at the beginning of the installation process.

10.1 – Define Connector for ACF2 Started Tasks in ACF2

Started task	Description
CTSGATE	Connector for ACF2 Gateway
CTSACS	Connector Transaction Server (CS)
CTSACD	Connector Notification Server (CD)
CTSAONI	Connector Online Interceptor
CTSAOFI	Connector Offline Interceptor

> **Note**
> In the list of started tasks used in this section, it is assumed that the default value CTS was accepted for the DEFPARMS parameter PROCPREFS. If you assigned a different value to this parameter, modify the started task names accordingly.

- Started tasks CTSACS, CTSACD, CTSAONI, and CTSAOFI must have AUDIT, MUSASS and MUSUPDT privileges defined in the logonid record. For additional privileges that must be assigned to support any user-defined fields, see Support User-Defined Fields in the Logonid Record.
- When you have any user-defined field (USERCFDE) with privileges specified for the authorization operands LIST or ALTER (for example LIST=SECURITY+ACCOUNT), then select one of these privileges and assign it to all of the following Connector for ACF2 started tasks: CTSACS, CTSACD, CTSAONI, and CTSAOFL.

  For example, when a user-defined field shows LIST=SECURITY+ACCOUNT, then CTSACS, CTSACD, CTSAONI, and CTSAOFL must be assigned either the ACCOUNT or the SECURITY privilege.
- Any user-defined field in ACF2 with privileges not assigned to Connector for ACF2 started tasks are not updated in the IdentityIQ database. In the IdentityIQ GUI, these fields will appear empty.

10.2 – Set Permissions to Connector Datasets

Permit READ access for the Connector DIAGLVL and CLIST libraries for your MVS system programmers, z/OS staff, or SailPoint Mainframe support team who should be able to see them.

> **Important**
> Do not allow users access to any DIAGLVL or CLIST libraries in the CTSACF2 installation job.

Permit all Connector for ACF2 installation libraries to be accessed by Connector for ACF2 started tasks listed above with read and write authorizations.

10.3 – Protect the Encryption Keys Datasets

Transmitted Data Encryption Keys Dataset

> **Note**
> This permission is only required when Transmitted Data Encryption is implemented.

Set ACF2 to permit only Connector for ACF2 servers (CTSACS and CTSACD) READ access to the encryption key dataset ENCREXT created in Procedure "*9.4 – Set up Secured Communication*" in 9 – Customize Communication Settings. No other accounts, other than the installer User ID, must be authorized to access this dataset (not even READ authorization).

Stored Data Encryption Keys Dataset

Set ACF2to permit only Connector for ACF2 servers (CTSACS and CTSACD) and Connector Interceptors (CTSAONI and CTSAOFI) READ access to the encryption key dataset ENCRINT created in 8 – Format Connector for ACF2 Datasetsprocedure. No other accounts, other than the installer User ID must be authorized to access this dataset (not even READ authorization).

10.4 – Define an OMVS Segment

Define an OMVS segment for the user ID and group ID associated with the CTSGATE started task. For more information, see details provided within the CTSACF2 member.

10.5 – Grant CTSGATE with Authority to Use TCP/IP Stack

This permission is required only when ACF2 SERVAUTH resource class is defined to protect TCP/IP resources from unauthorized access. For more information, see details provided within the CTSACF2 member.

11 – Adjust for Longer Managed System Names

The name assigned to the Managed System may be up to 32 characters long. However, if the Managed System name length is greater than eight characters, certain adjustments must be performed.

When the length of the Managed System name is greater than eight characters, do the following after completing the installation of Connector for ACF2:

1. Select a short name (up to eight characters) for the Managed System name (referred to as the *short MS name*). This name must be unique for each Managed System managed by the instance of Connector for ACF2.
2. Edit the RSSPARM member in the Connector PARM library. The value for parameter RSS_WORK_DIR contains the prefix for dynamically allocated datasets. The prefix contains the Managed System name as a qualifier. Change the Managed System name to the short Managed System name so that the value conforms to MVS dataset naming conventions.
3. Edit the member CTSOFLI in Connector JCL library. The member contains DELETE and LISTCAT commands regarding Offline Interceptor datasets. The dataset names contain the Managed System name as a qualifier. Change the Managed System name to the short Managed System name so that the dataset names conform to MVS dataset naming conventions.
4. Edit members CTSOFLMI and CTSOFLMR in Connector for ACF2 JCL library. Add the parameter RSSQ= with the value used as the short Managed System name to the EXEC statement in both members. Save the members.

> **Note**
> The short RSSNAME qualifier must be identical in all modifications.

12 – Adjusting Managed System Administrator Attributes

12.1 - Provide Managed System Administrator Passwords

This section is relevant when you define a new Application or when you set a new Managed System Administrator in SailPoint.

ACF2 Connector does not save the Managed System Administrator passwords in a file, as most of the other Connectors do.

When a new Managed System Administrator is defined in **Application** definition in IdentityIQ or in **Source** definition in IdentityNow, the Managed System Administrator password or phrase is required for verification, unless a protected user is defined as the Managed System Administrator. The Managed System Administrator User ID and password or phrase are sent from SailPoint to the ACF2 Connector where the password or phrase is verified. After verification, the password or phrase is not saved anywhere on the Connector's platform, or in SailPoint.

Before setting the managed System Administrator user and password or phrase in SailPoint, ensure that the password or phrase of the user is not expired (a new password or phrase set using the INSERT or CHANGE command is usually expired automatically, and must be changed when logging on for the first time). If the password or phrase is expired, password or phrase verification done by ACF2 Connector will fail.

12.2 - Verify Managed System Administrator Permissions

Set operations, activated from SailPoint, are done in ACF2 under the Managed System Administrator User ID. Therefore, this User ID requires SECURITY+ACCOUNT privileges in ACF2.

13 – Add Connector for ACF2 Libraries to the MVS Authorized Libraries List

1. Add the Connector LOAD library and Connector CTRANS library to the MVS APF authorized libraries list.
2. Edit member PROG*nn* in the SYS1.PARMLIB library. Add the Connector LOAD and CTRANS libraries and their volumes to the list using the APF statement.
3. Add the libraries to the active APF list by specifying the following operator command. Replace *nn* with the suffix of the PROG*nn* member that is updated:

```
SET PROG=nn
```

14 – Review the Installation

You have completed the Connector for ACF2 installation procedure. Before starting Connector for ACF2, be sure to the installation checklist and review the installation procedures to make sure none were omitted.

14.1 – Verify the ACF2 Connector Gateway Installation

Verify that the Connector for ACF2 Gateway is installed properly by issuing the following operator command:

```
S CTSGATE
```

The Connector for ACF2 Gateway starts, and then automatically starts the Connector servers: Transaction Server (CS) and Notification Server (CD). The Connector for ACF2 Gateway waits for communication to be established with SailPoint.

> **Note**
> For more information about starting and stopping Connector processes, see Operations.

14.2 – Verify that the ACF2 Connector Communicates with SailPoint

Perform the required configuration activities in Connector Gateway and SailPoint.

For more information, see Communication Parameters Coordination and the appropriate Connector Gateway, SailPoint guides for detailed configuration description.

Verify that all components communicate successfully.

> **Note**
> For IdentityIQ, perform the **Test Connection** to synchronize the keywords file, perform account/group aggregation, and so on.

14.3 – Verify ACF2 Connector Interface

To test Connector for ACF2 without involving SailPoint, see Appendix F: Connector for ACF2 Batch Utility.

> **Caution**
> It is not recommended to perform LISTUSER to all accounts with all attributes as this produces a very large `sysout` file.

15 – (Optional) Migrate Local Changes

Local changes in the previous version of ACF2 Connector environment may be copied manually to new ACF2 Connector environment.

Such local changes may refer to:

- Parameters set in RSSPARM member in PARM library.
- RSSAPI member in PARM library.
- Scripts in **USER.CLIST** library.

> **Note**
> If there are parameters set in CTSPARM member in PARM library of the previous version and there is a need to migrate them to the current version, contact SailPoint Customer Support.

16 – Configure Automated Startup of Connector for ACF2

If you use an automatic startup tool to start your required applications after system initialization, add CTSGATE to the definitions. Otherwise, add the following command to member COMMND*nn* in SYS1.PARMLIB to start Connector for ACF2:

```
S CTSGATE
```

> **Note**
> If the Connector for ACF2 will be started automatically during system initialization, be sure it starts after TCP/IP and the ACF2 subsystem complete their initialization.

17 – Customizing ACF2 Support

Customization of ACF2 support must be performed at this time. For information and step by-step instructions, see ACF2 Support Customization. Perform this customization before continuing to the next procedure in the installation.

18 – Post Installation Checks

Performing any operation performs the Test Connection and Keywords synchronization as the first operation and it would update keywords on Mainframe Connector in **RSSKWDS** file.

1. Start the Mainframe Connector.
2. Start Connector Gateway and ensure that in Mainframe **CTSGATE** a connection is established with the Connector Gateway.
3. To synchronize keywords after Mainframe Connector installation, check application schema of configured ACF2 application and perform **Test Connection**.
4. For any existing application, check application schema of configured ACF2 application.
5. To use a new schema attribute, add it in to account or group schema attribute accordingly.

> **Important**
> If you install a new Mainframe Connector retaining the same IdentityIQ application, then there may be issue of keywords not being synchronized. In this case update the descrip-

> tion of any schema attribute for any minor changes.

6. Save the application.

Uninstalling the Connector

To uninstall the Connector for ACF2, perform all of the following procedures.

1. Delete all Connector datasets.
2. Delete all Connector procedures from the system PROCLIB.
3. Remove references to Connector LOAD and CTRANS libraries from PROGxx member in SYS1.PARMLIB.
4. If STCJOBs are installed delete them from the system STCJOBs library.
5. If you configured Connector for automatic start, remove the relevant definitions.
    - Security definition for Connector started tasks
    - Connector dataset permissions
    - ACF2 accounts defined for Connector

ACF2 Support Customization

The following topics are discussed in this chapter:

> **Important**
> It is highly recommended that you review all of the topics discussed in this chapter to determine which are relevant for your implementation of Connector for ACF2.

Overview of Connector for ACF2 Interceptors 41
Set Logging Options 46
Install and Customize Online Interceptor 46
Install Offline Interceptor Interface 55
Shared ACF2 Database Support 56
Support User-Defined Fields in the Logonid Record 60
Rule Set Backup 61
Supporting Mixed Case Passwords 62
Defining and Using SailPoint Connector for ACF2 Groups 62

Overview of Connector for ACF2 Interceptors

This section provides background material relevant to several of the customization procedure described in this chapter.

The Connector for ACF2 detects changes and events in ACF2 via the following components:

- Online Interceptor (CTSAONI)
- Offline Interceptor (CTSAOFI)

These Connector for ACF2 components, described below, process data in cooperation with ACF2 and various components within your z/OS system. This chapter describes the required customization of these components.

> **Note**
> Online Interceptor and Offline Interceptor are not supported for IdentityNow.

Online Interceptor Logic

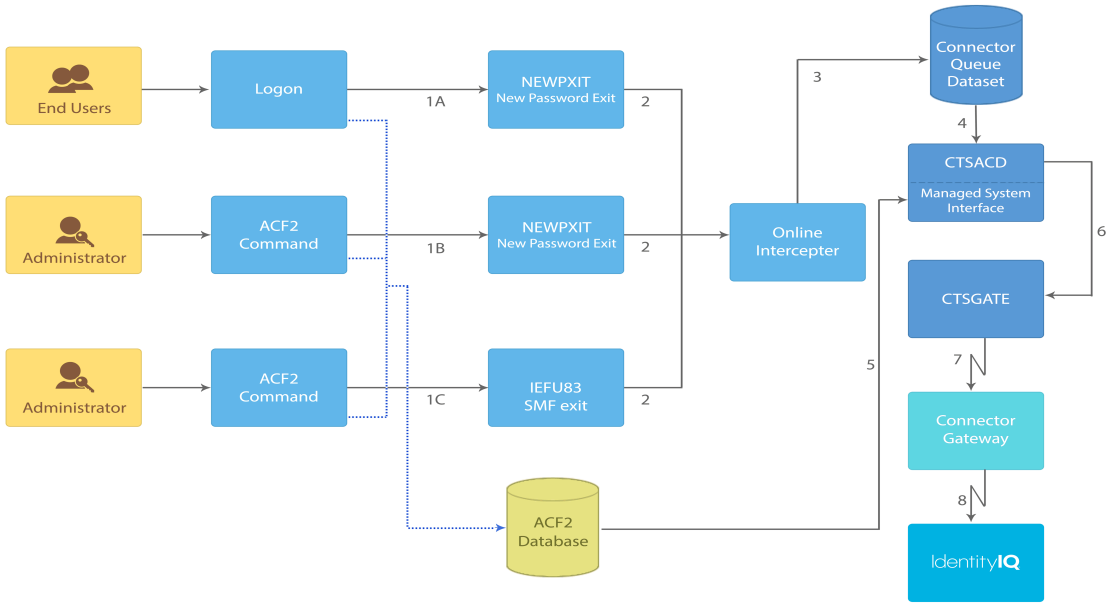
The Connector Online Interceptor detects, in real-time, ACF2 administration events that occur on the platform, and records them so that they can be reported to IdentityIQ. To accomplish this, the SMF record exit (IEFU83) is used to intercept every ACF2 command issued in the system, and transfer information regarding the commands to the Online Interceptor.

If password synchronization support is required, the Online Interceptor notifies IdentityIQ of password change events made by ACF2 accounts. To intercept password change events, ACF2 postinit exit ICHRIX02 and ACF2 new password exit ICHPWX01 are used. Both exits must be installed to intercept all password change events in the system.

When a system exit intercepts an event, it notifies the Online Interceptor started task via cross-memory services that the event has been intercepted. The Online Interceptor records the event in Connector for ACF2 datasets. The data are then reported to IdentityIQ by the Connector Notification Server (CD), via CTSGATE.

As long as the Online Interceptor is active in the system, ACF2 events and changes are recorded, even if the Connector for ACF2 is inactive. When Connector for ACF2 is restarted, the recorded data are transmitted to IdentityIQ.

The processing flow of the Connector Online Interceptor is illustrated in the following figure:



The Connector Online Interceptor detects ACF2 events in one of the following manners:

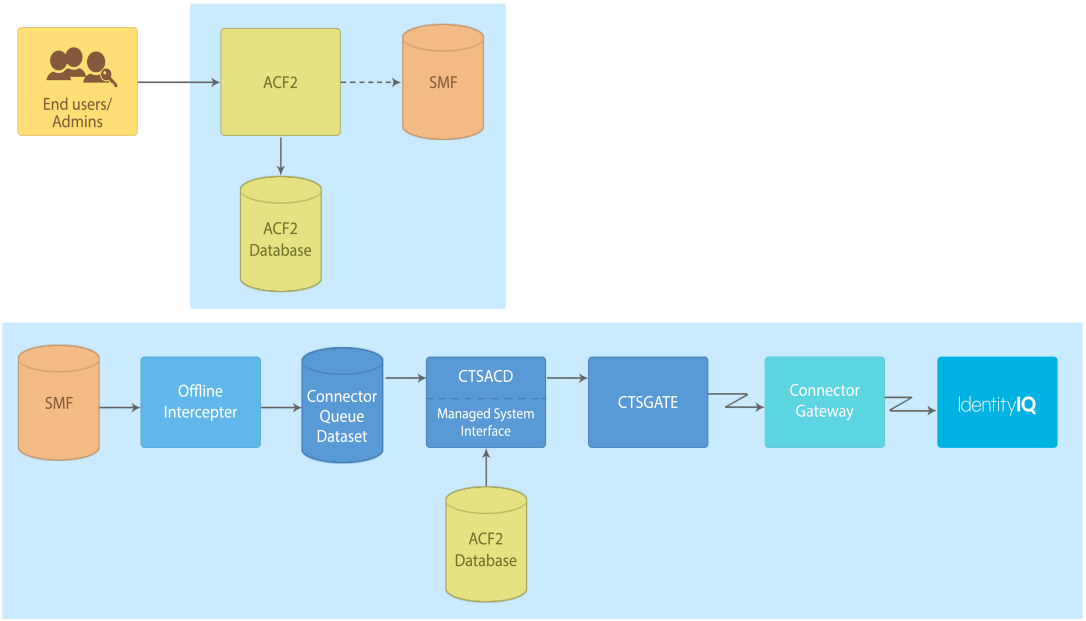
- When a ACF2 user changes his/her password during the logon process or by an ACF2 command, ACF2 calls exit NEWPXIT [1A].
- When ACF2 administrators issue an ACF2 command, ACF2 calls SMF exit IEFU83 to log the command [1C].

- When a ACF2 administrator issues an ACF2 command that changes a user's password, ACF2 call both exit NEWPXIT [1B] and SMF exit IEFU83 [1C]. In this case, 2 events will reach the Online Interceptor.

In any of these situations, the exit that intercepts the event passes the event to the Connector Online Interceptor via cross-memory services [2]. The Online Interceptor then writes the event to the Connector QUEUE dataset [3]. The Connector Notification Server (CD) reads an event from the QUEUE dataset [4]. If SMF event it reads the updated account from ACF2 database [5]. In any event, SMF exit or Password exit it transfers the event to the Connector for ACF2 Gateway (CTSGATE) [6] which transfers the event to the Connector Gateway [7] which passes it to SailPoint [8].

Offline Interceptor Logic

The Offline Interceptor is intended to be used if the Online Interceptor is not installed or following a period of Online Interceptor inactivity. While the Online Interceptor intercepts ACF2 administration events as they occur, the Offline Interceptor reads an SMF dataset in which ACF2 has logged events that have already occurred. The process flow of the Connector Offline Interceptor is illustrated in the following figure:



When a user issues a ACF2 command, ACF2 calls SMF. SMF logs events to SMF dataset SYS1.MANx. Utility IFASMFDP reads dataset SYS1.MANx and writes the event record to a historical SMF dataset. The Offline Interceptor then reads the historical SMF dataset and writes the events to the Connector QUEUE dataset. The Connector Notification Server (CD) reads the QUEUE dataset, gets from ACF2 DB the up-to-date account record and transfers the event data to the Connector for ACF2 Gateway, which then transfers the event data to Connector Gateway which passes it to SailPoint.

> **Note**
> Password change events are not handled by the Offline Interceptor (If needed, the Online Interceptor must be used).

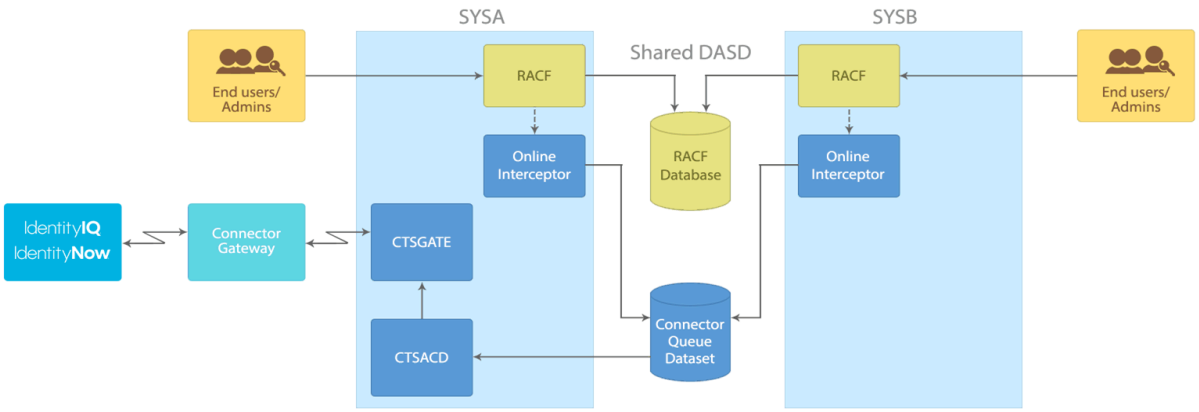
Interceptors in a Shared ACF2 Database Environment

A shared ACF2 database environment consists of two or more ACF2 systems which share the same ACF2 database.

In a shared database configuration, one of the systems runs a full instance of Connector for ACF2. This system is referred to as the primary system. Each additional ACF2 system sharing the database is referred to as a *secondary system*. Each secondary system only runs one component of Connector for ACF2 the Interceptor.

The primary system communicates with IdentityIQ and executes the transactions generated by IdentityIQ (in the regular manner). The secondary systems runs the Connector Interceptors to ensure that any updates made to the ACF2 database from these systems are propagated to IdentityIQ. In this way, the IdentityIQ and ACF2 databases are kept synchronized.

For example, assume that systems SYSA and SYSB share the ACF2 database. SYSA is the primary system which runs the complete Connector for ACF2. SYSB is the secondary system which runs only the Online Interceptor. This configuration is shown in the following figure:



When a transaction is issued by IdentityIQ, it is executed by the Connector Transaction Server (CS) running on the primary system, and the appropriate updates are made to the ACF2 database.

When a local update is done in either SYSA or SYSB, it is intercepted by the appropriate Interceptor running in the system and written to the Connector QUEUE dataset.

The Notification Server (CD) running in the primary system reads the local update from the Connector QUEUE dataset and forwards it to Connector for ACF2 Gateway which passes it to IdentityIQ. This ensures that the IdentityIQ and ACF2 databases are kept synchronized.

User-Defined Fields in the Logonid Record

ACF2 supports the definition of user-defined fields in the logonid record. These fields can be supported by Connector for ACF2. For more information, see Assign New Required Privileges to ACF2 Connector.

To support a user-defined field in the logonid record, a corresponding attribute must be defined in Application's schema in SailPoint. Each attribute must have the same name as its corresponding logonid record field for Connector for ACF2 to support it.

Once the attribute is defined, Connector for ACF2 handles it in the same manner as predefined attributes.

The Connector for ACF2 analyzes the ACFFDR module and automatically detects user-defined attributes with names that match logonid record fields. Once such fields are identified, they are retrieved and can be managed in the same manner as predefined attributes.

Specifying user-defined specific fields is not mandatory at installation time and can be done later. However, additional accounts aggregation will be required to fill the user-defined attributes with their values.

> **Note**
> For more information, see Support User-Defined Fields in the Logonid Record .

Rule Set Backup

By default, Connector for ACF2 backs up every access/resource rule set before it is modified or deleted from IdentityIQ.

The backup is performed by calling the REXX CTSBKRUL script from the following Connector for ACF2 functions, before the rule set modification:

- UPDRES (Update Resource)
- DELRES (Delete Resource)
- ADDACE (Add ACE)
- UPDACE (Update ACE)
- DELACE (Delete ACE)

Script CTSBKRUL resides in the CLIST dataset of the Connector for ACF2 installation. When executed, CTSBKRUL creates a new partitioned dataset in which it stores a decompiled source of the rule set.

By default, the rule set backup is active. You can choose to deactivate it or to modify script CTSBKRUL to match your site standards.

Set Logging Options

The Connector Interceptors detect changes made to security definitions and record these changes so that they can be reported to IdentityIQ. To detect these changes, ACF2 should be set to log all ACF2 commands that are issued in the system, and SMF should be set to collect these records.

For more information regarding Connector Interceptors, see Overview of Connector for ACF2 Interceptors.

Set SMF Parameters

ACF2 generates audit information regarding ACF2 commands in SMF record type 230. These records are passed to the SMF record exit (IEFU83), which transfers information regarding the commands to the Online Interceptor.

To ensure that these records are collected for future processing and are passed to SMF exit IEFU83, SMF parameters must be set to collect generated records.

1. Check ACF2 settings.

   Use command ACF (sub-command SHOW SYSTEM) to find out which SMF records ACF2 uses to log events.

2. Edit the SMFPRMnn member.

   Edit the SMFPRMnn member in your SYS1.PARMLIB library. This member specifies which SMF records are collected by SMF.

   nn is the number specified in member IEASYS in SYS1.PARMLIB, or **00**, if it was not specified.

   Make sure parameter TYPE specifies that record type used by ACF2 (default 230) is collected for all sub-systems.

   For more information on SMFPRMnn parameters, refer to the MVS/ESA Initialization and Tuning Guide.

   Save the member (if it was modified).

3. Activate SMF parameters.

   If you modified member SMFPRMnn, refresh SMF definitions by issuing the following operator command:

   ```
   SET SMF=nn
   ```

   where *nn* is the suffix of the SMFPRMnn member that was updated.

Install and Customize Online Interceptor

The "Exits" must be installed in the system as required:

1. If local changes in the ACF2 DB are to be updated in real time in IdentityIQ then **SMF exit IEFU83**, provided with ACF2 Connector, must be installed.

2. If password/phrase changes of ACF2 accounts are to be updated in real time in IdentityIQ then **ACF2 new password exit** (NEWPXIT), provided with ACF2 Connector, must be installed.

Exits

- **SMF record exit IEFU83** – Intercepts ACF2 commands and transfers information to the Connector Online Interceptor regarding each command.
- **ACF2 new password exit** – Intercept password changes and transfers password data to the Connector Online Interceptor regarding each password change.

> **Note**
> The usage of these exits is described in further detail under Online Interceptor Logic.

1 – Install SMF Exit IEFU83

To install SMF exit IEFU83, perform the following:

1. Create the CTSU83A exit module.

   Member ASMU83A in the Connector INSTALL library is a sample member to assemble and link Connector for ACF2 exit IEFU83. ACFU83A member in INSTALL library is the source code of the IEFU83 exit.

   Review the JCL carefully and submit the job to create the CTSU83A load module in the Connector LOAD library. All job steps must end with a condition code of `0`, except for the following:

   - CHECRCF – Ends with the condition code `4`.
   - RCFU83A – This step is not executed.

2. You can either copy CTSU83A module to your LPALIB (this requires you to do an IPL) or you can use the dynamic exit facility feature of MVS. It is recommended that you use the dynamic exit facility. The following step assumes that you use the dynamic exit facility for the SMF exit. If, for any reason, you want to set the SMF exit in the LPA (and do an IPL) skip to step 3.

   a. Define the SMF exit in the MVS dynamic exits facility.

      Edit member PROG*nn* in SYS1.PARMLIB to define the new SMF exit module, for all SMF sub-systems.

      Add the following statements to the member:

      ```
      EXIT ADD EXITNAME(SYS.IEFU83) MODNAME(CTSU83A)DSNAME (<CTSA LOAD library>)
      ```

```
EXIT ADD EXITNAME(SYSSTC.IEFU83) MODNAME(CTSU83A)
DSNAME (<CTSA LOAD library>)
EXIT ADD EXITNAME(SYSTSO.IEFU83) MODNAME(CTSU83A)
DSNAME (<CTSA LOAD library>)
```

If JES3 is active in your environment, please add the following line as well:

```
EXIT ADD EXITNAME(SYSJES3.IEFU83) MODNAME(CTSU83A)
```

b. Activate the exit by issuing the following operator command:

```
SET PROG=nn
```

where *nn* is the suffix of the PROG*nn* member which was updated in step 2 above.

> **Note**
> Installation of SMF exit IEFU83 is complete. Proceed to step 3 if you need to set the SMF exit in the LPA and do an IPL.

3. To set the SMF exit in the LPA (and do an IPL):

a. Copy CTSU83A to your LPA system library.

Edit member CPYU83A in the Connector INSTALL library. The job copies load module CTSU83A to your system LPA library.

Review the job carefully and submit the job. All job steps must end with a condition code of 0.

b. Activate the exit in LPA.

Once the exit module resides in your LPA library, activate the module. This is usually performed by the next IPL with the CLPA option.

If you have a product which can dynamically load modules to the LPA (for example, RESOLVE, LOOK, OMEGAMON) you can use it to add the exit to the LPA. However, when loaded in this manner, the exit remains active only until the next IPL.

> **Note**
> Installation of SMF exit IEFU83 is complete.

2 – Install ACF2 New Password Exit (NEWPXIT)

> **Note**
> Install the new password exit only if support for password synchronization is required.

The ACF2 new password exit can be installed using one of the following methods:

- Online Interceptor loads password exit
- ACF2 loads password exit

Each method is described in more detail in the following sections.

Method 1 – Online Interceptor Loads Password Exit

When the Online Interceptor task is started, it replaces any existing site password exit and keeps the address of the site exit. When the Connector for ACF2 password exit is called, it will first do its own processing, and then call the site exit, based upon the address saved at startup.

The module name which functions as the password exit and is called by the Online Interceptor is CTS2NPX.

> **Note**
>
> - When multiple password exits exist and problems such as loop or abends occur, SailPoint recommends that you use the "ACF2 loads password exit" method (described below).
> - If two or more instances of Connector for ACF2 on the same machine use the dynamic installation of the new password exit, restrictions apply to the order in which the Online Interceptors for each instance of Connector for ACF2 are shut down. For more information, see Starting and Stopping the Online Interceptor.
> - If you issue the command `F ACF2,REFRESH`, the Connector for ACF2 password exit will be replaced by the default password exit. If the above command is issued, the Online Interceptor must be recycled to re-install the password exit.

To set up the Online Interceptor to load the ACF2 new password exit, perform the following:

1. Edit member RSSPARM in the Connector PARM library.
2. The parameter ONLI_DYNAM_NPX defines whether the Connector Online Interceptor will dynamically install NEWPXIT when it is started.

   Set the ONLI_DYNAM_NPX parameter to **Y**.
3. Save the member

> **Note**
> Dynamic installation of the ACF2 new password exit is complete. Exit NEWPXIT is dynamically installed when the Online Interceptor is started.

Method 2 – ACF2 Loads Password Exit

With this method the password exit load module is set in the ACF2 LPA library.

The exit is loaded using any of these methods:

- IPL
- z/OS SETPROG LPA command
- Any product that can dynamically load modules to the LPA

The following table lists the members in the INSTALL library which are involved in this procedure.

Member	Description
CTSNPXA	Connector for ACF2 new password exit source code.
CTSNPXB	Connector for ACF2 sample exits driver source code.
ASMNPXA	Sample job to compile and link CTSNPXA.
ASMNPXB	Sample job to compile and link CTSNPXB.
CPYNPXA	Sample job to copy CTSNPXA module to the system LPA library as ACFPNXIT.
CPYNPXB	Sample job to copy CTSNPXB module to the system LPA library as ACFNPXIT.

Before installing the exit:

1. Make sure the NEWPXIT of the ACF2 GSO refers to the default name, which is ACFNPXIT. If not, change it using the command:

   ```
   CHANGE EXITS NEWPXIT(ACFNPXIT)
   ```

2. Next, refresh the GSO in ACF2.
3. Make sure that the ONLI_DYNAM_NPX parameter is left in its default N setting.
4. Refer also to the *eTrust ACF2 Security for z/OS Administrator Guide*, to the paragraph "eTrust ACF2 Exit Specifications (EXITS)." As specified there, when running a release of the operating system that supports the z/OS SETPROG LPA command, you can use this instead of an IPL, after a new copy of the password exit is set.
5. Before proceeding with the installation, you must decide upon which type of installation is relevant for your site:

- **Single Exit Installation**: For sites that do not have a ACF2 new password exit installed.
- **Multiple Exit Driver Installation**: For sites that already have a ACF2 new password exit installed. The multiple exit driver internally defines the exits that are supported and calls each exit in turn.

To perform the installation of ACF2 exit NEWPXIT when using a single exit, perform the following:

1. Create the CTSNPXA exit module.

   Member ASMNPXA in the Connector INSTALL library is a sample member used to assemble and link Connector for ACF2 password exit. A sample of a password exit is available in member CTSNPXA in INSTALL library.

   Review the JCL carefully and submit the job to create the CTSNPXA load module. All job steps must end with a condition code of `0`.

2. Copy CTSNPXA to your LPA system library.

   Edit member CPYNPXA in the Connector INSTALL library. The job copies CTSNPXA to your system LPA library as ACFNPXIT.

   > **Note**
   > Any previous copy of exit ACFNPXIT in the LPA library is overwritten by the job. It is recommended that you create a backup copy of each previous exit.

   Review the jobs carefully and submit the jobs. All job steps must end with a condition code of `0`.

3. Activate the exits in LPA.

   Once the exit modules reside in your LPA library, they must be activated. This is performed by the next IPL with the CLPA option, with any product that can dynamically load modules to the LPA, or with the z/OS SETPROG LPA command together with ACF2 command to refresh the exit:

   ```
   SETPROG LPA,ADD,MODNAME=(abcdefg),DSNAME=cai.cax1lpa F ACF2,REFRESH(EXITS)
   ```

   For further description, refer to the *eTrust ACF2 Security for z/OS Administrator Guide*, to the paragraph "eTrust ACF2 Exit Specifications (EXITS)."

   > **Note**
   > Installation of CA-ACF2 exit NEWPXIT using a single exit without SMP/E is complete.

To perform the installation of ACF2 exit NEWPXIT when using the multiple exit driver, perform the following:

1. Create the CTSNPXA exit module.

   Member ASMNPXA in the Connector INSTALL library is a sample member used to assemble and link Connector for ACF2 password exit. A sample of password exit is available in member CTSNPXA in INSTALL library.

   Review the JCL carefully and submit the job to create the CTSNPXA load module. All job steps must end with a condition code of `0`.

2. Customize the exit table for the multiple exit driver.

   Edit member CTSNPXB in the Connector INSTALL library. Locate the exit table at the end of the member and customize it to call your local exit.

   Exits are defined in an exit table in the source code of the multiple exit driver. All the exits in the table are called, regardless of their return code. The multiple exit driver returns the highest return code of all the exits that were called.

   Save the member.

3. Create the CTSNPXB load module.

   Member ASMNPXB in the Connector INSTALL library is a sample member used to assemble and link the Connector for ACF2 sample driver for multiple password exits.

   > **Note**
   > You must add an INCLUDE statement to the linkage editor input stream so that the exit driver is linked with your local exit.

   Review the JCL carefully and submit the job to create the CTSNPXB load module in the Connector LOAD library. All job steps must end with a condition code of `0`.

4. Copy CTSNPXB to your LPA system library.

   Edit member CPYNPXB in the Connector INSTALL library. The job copies CTSNPXB to your system LPA library as ACFNPXIT.

   > **Note**
   > Any previous copy of ACFPNXIT in the LPA library is overwritten by the job. It is recommended that you create a backup copy of the previous exit.

   Review the job carefully and submit the job. All job steps must end with a condition code of `0`.

5. Activate the exits in LPA.

Once the exit modules reside in your LPA library, they must be activated. This is performed by the next IPL with the CLPA option, or with any product that can dynamically load modules to the LPA, or with z/OS SETPROG LPA command together with ACF2 command to refresh the exit:

```
SETPROG LPA,ADD,MODNAME=(abcdefg),DSNAME=cai.cax1lpa F ACF2,REFRESH(EXITS)
```

For further description, refer to the *eTrust ACF2 Security for z/OS Administrator Guide*, to the paragraph "eTrust ACF2 Exit Specifications (EXITS)."

> **Note**
> Installation of CA-ACF2 exit NEWPXIT using the multiple exit driver without SMP/E is complete.

3 - Verify that the Online Interceptor is Installed Properly

Verify that the Online Interceptor is installed properly by issuing the following operator command:

```
S CTSAONI
```

The Online Interceptor starts and begins recording ACF2 events to the ACF2 Connector Queue dataset.

4 - Update IdentityIQ Definitions

IdentityIQ definitions must be updated to allow asynchronous events to be sent from ACF2 connector to IdentityIQ.

For more information, see "Settings for configuring password Interceptor and Online Interceptor" section of *SailPoint Quick Reference Guide for Gateway Connectors*.

5 - Configure Automated Startup of Online Interceptor

Make the appropriate changes to the system startup procedures to start Connector for ACF2 Online Interceptor automatically after IPL.

If an automatic startup tool is used to start all the required applications after system initialization, add CTSAONI to this automatic startup tool definitions.

Otherwise, add the following command to COMMNDnn member in SYS1.PARMLIB to start the Online Interceptor after IPL:

```
S CTSAONI
```

6 - (*Optional*) Controlling Online Interceptor Memory Consumption

You can control memory size used by **Online Interceptor**. This can be achieved by setting the optional parameter ONLI_MAX_EVENTS in the RSSPARM member in the PARM dataset.

ONLI_MAX_EVENTS set the maximum number of events that the Online Interceptor can accumulate in memory. ONLI_MAX_EVENTS is optional. If this parameter does not exist in the RSSPARM member, the default value of 20,000 events is used.

The minimum value for ONLI_MAX_EVENTS is 2,000. Each entry occupies 2,560 bytes in memory, depending on the event type (user, group, connection, password) and length of userid, group, password.

Use the following syntax for this parameter:

```
managedSystemName     ONLI_MAX_EVENTS     numberOfEvents
```

If this parameter is not specified, the default value of 20,000 is used (that is, the events can occupy up to about 50 MB in the Online Interceptor's address space).

The optional parameter ONLI_MIN_NOTIFY_EVENT% sets a threshold for issuing a warning message that the memory for accumulating event may soon be entirely filled. When the stated percentage of available memory for accumulating events is filled, message CTS4509W (described below) is issued.

Use the following syntax for this parameter:

```
managedSystemName     ONLI_MIN_NOTIFY_EVENT%     value%
```

> **Note**
> The value specified must be followed by the % symbol. If the ONLI_MIN_NOTIFY_EVENT% parameter is not specified in the RSSPARM member, the default value of 10% is used.

For example, given the following settings:

```
RSS1      ONLI_MAX_EVENTS             5000
RSS1      ONLI_MIN_NOTIFY_EVENT%      20%
```

You can expect the following:

- Space for 5000 intercepted events is allocated in memory (approximately 12.5 MB).
- Message CTS4509W will be issued when only 20% of the allocated space remains (that is, when more than 4,000 events reside in memory).

7 - (*Optional*) Configure Event Interception Filtering

Event Interception filtering enables you to control which entity events intercepted in the Managed System by the Online Interceptor are reported to IdentityIQ.

The following RSSPARM parameters enable you to filter the interception of events on the Managed System:

- **ONLI_EVENT_GROUP** – Specifies whether the Online Interceptor should report group events to IdentityIQ.

  Possible values for this parameter are Y or N.

- **ONLI_EVENT_USER** – Specifies whether the Online Interceptor should report user events to IdentityIQ.

  Possible values for this parameter are Y or N.

- **ONLI_EVENT_USER_PWD_ONLY** – Specifies which intercepted user events are reported to IdentityIQ.

  > **Note**
  > This parameter is only relevant when ONLI_EVENT_USER is specified as **Y**.

  Possible values for this parameter are:

  - **Y** – Only user password change events are reported to IdentityIQ
  - **N** – (Default) All intercepted user events are reported to IdentityIQ.

To filter the interception of Managed System events:

1. Stop Connector for ACF2 Online Interceptor.

   For more information, see Starting and Stopping the Online Interceptor.

2. Edit the RSSPARM member in the Connector PARM library.

3. Insert or modify one or more of the following parameters as necessary.

```
rss_name ONLI_EVENT_USER_PWD_ONLY N <== Modify as required
rss_name ONLI_EVENT_USER Y <== Modify as required
rss_name ONLI_EVENT_GROUP Y <== Modify as required
```

4. Save the member.

5. Restart the Connector for ACF2 Online Interceptor.

   For more information, see Starting and Stopping the Online Interceptor.

Install Offline Interceptor Interface

The Connector for ACF2 Offline Interceptor is intended to be used following a period of time during which the Online Interceptor was not active, or if the Online Interceptor is not installed.

While the Online Interceptor intercepts ACF2 administration events as they occur, the Offline Interceptor reads SMF datasets in which ACF2 has logged events that have already occurred.

Before they are processed, SMF records in the SYS1.MANx datasets should be copied into a sequential dataset by SMF utility IFASMFDP.

JCL library member RCFAOFI includes a sample job for activating the program IFASMFDP. During activation of IFASMFDP, a sequential dataset is created that includes only SMF80 and SMF30 records. These datasets are necessary for the Offline Interceptor started task (CTSAOFS).

Most MVS sites dump the contents of the SYS1.MANx datasets to a sequential dataset as a step in a daily job or when the SYS1.MANx becomes full.

Examination of the SYS1.MANx dataset directly by the Offline Interceptor may not be useful since SYS1.MANx may have already been cleared. This could result in events being missed by the Offline Interceptor.

Therefore, you should integrate the Connector Offline Interceptor into the SMF processing jobs which clear the SYS1.MANx datasets at your site.

For more information about the Offline Interceptor, see Offline Interceptor Logic.

Integrate Offline Interceptor into SMF Processing Jobs

The Offline Interceptor must be activated as a step in the SMF datasets' processing jobs.

Member ACFAOFS in the Connector JCL library is a sample job that demonstrates the use of the CTSAOFI JCL procedure by a job. The procedure activates the Connector Offline Interceptor with an input dataset that is passed as a parameter to the procedure.

Add a step to the SMF processing job to activate JCL procedure CTSAOFI. The value of the SMFDATA parameter should be the name of the SMF dataset holding the records to be processed by the Offline Interceptor.

Member ACFSMFD includes an example for activating IFASMFDP. During activation of IFASMFDP a sequential dataset is created. The dataset includes only SMF type 230 records, which are necessary for the Offline interceptor started task (CTSAOFS).

The Connector Offline Interceptor processes SMF records generated by ACF2 (default is 230). Verify that records are collected in the previous job steps and that they are not filtered.

Shared ACF2 Database Support

A shared ACF2 database environment consists of two or more ACF2 systems which share the same ACF2 database.

In a shared database configuration, one of the systems runs a full instance of Connector for ACF2. This system is referred to as the primary system. Each additional ACF2 system sharing the database is referred to as a *secondary system*. Each secondary system only runs one component of Connector for ACF2 the Interceptor.

Install Interceptor on Secondary System

The secondary system can intercept changes and events in a system via one the following components:

- Online Interceptor (CTSAONI)
- Offline Interceptor (CTSAOFI)

After installing Connector for ACF2 on the primary system, you must install either the Online or Offline Interceptor on each additional secondary system which shares the ACF2 database.

Secondary System Online Interceptor

To install the Online Interceptor on a secondary system:

1. Define Connector QNAME to the ENQ manager.

   The Connector for ACF2 serializes access to its QUEUE dataset by issuing an ENQ with the SYSTEMS option.

   To allow synchronized update of the QUEUE dataset from multiple systems, you should update your external ENQ manager (for example, GRS or MIM) with the Connector QNAME to be propagated to all systems in the external ENQ manager complex.

   Another option is to let the Connector for ACF2 use the RNL=NO ENQ/DEQ parameter telling GRS to always propagate the ENQ to all systems in the GRS complex. This can be controlled using the ENQRNL parameter in member CTSPARM in Connector for ACF2 PARM library.

   The default QNAME is xxxASYNC where xxx is the value set for %PROCPREFS% in the DEFPARMS member in the INSTALL library. The default may be customized by changing the QNAME parameter in member CTSPARM in Connector PARM library.

   For explanations on how to change and activate these parameters, see CTSPARM – Assembler Format Parameters.

2. Share the datasets.

   Place the datasets listed in the following table on a shared DASD so that they can be shared by the primary and secondary systems.

Dataset Name	Description
prefix.version.QUEUE	Connector QUEUE dataset
prefix.version.LOAD	Connector Load library
prefix.version.CMSG	Connector Msgs library
prefix.version.PARM	Connector PARM library
prefix.version.DIAGLVL	Connector Diagnostics library
prefix.version.ENCRINT	Connector for ACF2 Stored Data Encryption dataset
prefix.version.ACF2GDB	Connector internal groups DB
prefix.version.ACF2UDB	Connector internal users DB

3. Copy CTSAONI procedure and, if needed, STCJOB:

   - If STCJOBS are not used, copy the CTSAONI JCL procedure from the system PROCLIB of the primary system to the system PROCLIB of the secondary system.

- If STCJOBS are used:
  - Copy the CTSAONI STCJOB member to the system STCJOBS library.
  - If the value LOCALCOPY is set for %PROLCIB% in the DEFPARMS member in the INSTALL library, make sure the secondary system has access to the Connector for ACF2 PROCLIB library. Otherwise, copy the CTSAONI JCL procedure from the system PROCLIB of the primary system to the system PROCLIB of the secondary system.

4. In the secondary system(s):
   a. Add Connector LOAD and CTRANS libraries to the APF authorized libraries list.
   b. Define the CTSAONI started task to ACF2.

      For more information, see *10.1 – Define Connector for ACF2 Started Tasks in ACF2* in 10 – Define Connector for ACF2 in ACF2.
   c. Set ACF2 and SMF logging options.

      For more information, see Set Logging Options.
   d. Install Online Interceptor system exits (**IEFU83** and **NEWPXIT**).

      For more information see Install and Customize Online Interceptor.
   e. Start the Online Interceptor.
   f. After verification, configure automated startup of the Online interceptor.

      For more information see 5 - Configure Automated Startup of Online Interceptor.

Secondary System Offline Interceptor

To install the Offline Interceptor on a secondary system:

1. Define the Connector QNAME to the ENQ manager.

   The Connector for ACF2 serializes access to its QUEUE dataset by issuing an ENQ with the SYSTEMS option.

   To allow synchronized updates of the QUEUE dataset from multiple systems, you may need to update your external ENQ manager (for example, GRS or MIM) with the Connector QNAME. This name must be propagated to all systems in the external ENQ manager complex.

Another option is to let the Connector for ACF2 use the RNL=NO ENQ/DEQ parameter telling GRS to always propagate the ENQ to all systems in the GRS complex. This can be controlled using the ENQRNL parameter in member CTSPARM in Connector for ACF2 PARM library.

The default QNAME used by Connector for ACF2 is CTSASYNC. The default may be customized by changing the QNAME parameter in member CTSPARM in Connector PARM library.

For explanations on how to change and activate these parameters, see CTSPARM – Assembler Format Parameters.

2. Share the datasets.

   Place the datasets listed in the following table on a shared DASD between the primary and the secondary system.

Dataset name	Description
prefix.version.QUEUE	Connector QUEUE dataset
prefix.version.LOAD	Connector Load library
prefix.version.CMSG	Connector Msgs library
prefix.version.PARM	Connector PARM library
prefix.version.DIAGLVL	Connector Diagnostics library
prefix.version.ENCRINT	Connector for ACF2 Stored Data Encryption dataset

3. Copy CTSAOFI procedure and, if needed, STCJOB:

   - If STCJOBS are not used, copy the CTSAOFI JCL procedure from the system PROCLIB of the primary system to the system PROLCIB of the secondary system.
   - If STCJOBS are used:
     - Copy the CTSAOFI STCJOB member to the system STCJOBS library.
     - If the value LOCALCOPY is set for %PROLCIB% in the DEFPARMS member in the INSTALL library, make sure the secondary system has access to the Connector for ACF2 PROCLIB library. Otherwise, copy the CTSAOFI JCL procedure from the system PROCLIB of the primary system to the system PROCLIB of the secondary system.

4. In the secondary system(s):

   1. Add the Connector LOAD library to the APF authorized libraries list. For more information, see 15 – (Optional) Migrate Local Changes.

2. Set ACF2 and SMF logging options.

   For more information, see Set Logging Options.

3. Install Offline Interceptor interface.

   For more information, see Install Offline Interceptor Interface.

Support User-Defined Fields in the Logonid Record

To support a user-defined field in the logonid record and to allow its management as part of the Account entity, a corresponding attributes must be defined in Application's schema in SailPoint. Each attribute must have the same name as its corresponding logonid record field in order to be supported by Connector for ACF2.

Once the attributes exist, Connector for ACF2 automatically detects them and retrieves the values of the corresponding logonid record fields. From that point of time, the fields can be managed from IdentityIQ.

Specifying user-defined specific fields is not mandatory at installation time and can be done later. However, additional accounts aggregation will be required to fill the user-defined attributes with their values.

For more information, see User-Defined Fields in the Logonid Record.

Create User-Defined Managed System-Specific Attributes

To support user-defined logonid record fields, a list of these fields should be constructed. Definitions of all the fields can be found in member USERCFDE of the ACF2 CAIMAC dataset. This member contains a @CFDE macro entry for each user-defined field. This entry begins with the name of the field, followed by a list of all its attributes. A typical @CFDE entry should look like this:

```
@CFDE ORG,USRORG,CHAR,ALTER=SECURITY+ACCOUNT,TRIM=NO,
LIST=ALL,FLAGS=NULL,GROUP=0,ZERO=YES
```

To establish support of user-defined fields manually, perform the following:

1. Scan the member. Create a list of names of all the user-defined fields to be supported by SailPoint as Account data.
2. Create SailPoint attributes for all the user-defined fields. The name of each attribute must match the corresponding field.

> **Note**
>
> A hyphen ("-") in a ACF2 logonid record field name should be replaced with an underscore character ("_") in the corresponding SailPoint attribute.
>
> For example, the logonid record field name *PSWD-VIO* should be represented by the SailPoint attribute *PSWD_VIO*.

Assign New Required Privileges to ACF2 Connector

If you have any user-defined fields with privileges not assigned to Connector for ACF2 started tasks, then you should select one of these privileges and assign it to the following Connector for ACF2 started tasks: CTSACS, CTSACD, CTSAONI, and CTSAOFI.

For example, if a user-defined field has **LIST = ACCOUNT+SECURITY**, then CTSACS, CTSACD, CTSAONI, and CTSAOFI must be assigned with ACCOUNT and SECURITY privileges.

Perform Aggregation

After creating all the user defined-specific attributes (see the Create User-Defined Managed System-Specific Attributes procedure), reload the fields so they reflect the values of the corresponding user-defined logonid record fields, by issuing account aggregation from SailPoint.

Rule Set Backup

By default, Connector for ACF2 backs up every access/resource rule set before it is modified or deleted from IdentityIQ. The backup is performed by calling the REXX CTSBKRUL script which resides in dataset CLIST in the Connector for ACF2 installation.

Activate and Deactivate Rule Set Backup

By default, the rule set backup script is always activated by the various Connector for ACF2 resource and ACL functions. The script can be deactivated by specifying the following line in the RSSPARM file of the Connector for ACF2 installation, residing in the PARM library:

```
<rss_name> RULE_BACKUP N
```

where `<rss_name>` is the name of the ACF2 Managed System in which you wish to deactivate rule set backup.

Customize Backup Dataset Name

When executed, CTSBKRUL creates a new partitioned dataset in which it stores a decompiled source of the rule set before the rule set is modified.

The default name of the backup dataset is:

```
<ADMIN>.RULEBKUP.<RESTYPE>.D<DATE>.T<TIME>
```

where:

Parameter	Description
<ADMIN>	Name of the IdentityIQ administrator.
<RESTYPE>	Resource type (three characters).

Parameter	Description
<DATE>	Date in the format YYMMDD.
<TIME>	Time in the format HHMMSS.

To use this filename convention, a catalog alias must exist for the SailPoint administrator name so that datasets beginning with the <ADMIN> qualifier can be allocated.

You can edit the script and modify its filename convention as long as the filename convention is valid. To do that, enter the CLIST dataset of the Connector for ACF2 installation, and select member CTSBKRUL.

> **Note**
> When modifying the script code since a failure in its execution will lead to the failure of the calling Connector for ACF2 function.

Supporting Mixed Case Passwords

If mixed case is used in your ACF2 deployment, update the ONLI_PASSWORD_CASE parameter in PARM library in RSSPARM member to be **ASIS** or **UPPER** instead of the default **LOWER** value. When mixed case is used in ACF2 and ASIS or UPPER is set in RSSPARM, the passwords are sent to IdentityIQ without any translation by ACF2 Connector.

> **Note**
> ASIS and UPPER have same effect , which means no translation is done on the password.

Defining and Using SailPoint Connector for ACF2 Groups

For more information on ACF2 Connector implementation of groups, see ACF2 Group Representation.

If you decide to implement groups using SailPoint Connector for ACF2, you may either define the groups in SailPoint or use the Batch Utility to define these groups. before performing full aggregation of accounts and groups.

Each group is set and identified by the UIDMASK it has.

Secured Communication

Secured communication has the following aspects:

- Communication security using TLS or using Transmitted Data Encryption
- IP List validation - which allows control of the IP addresses which are allow accessing the ACF2 Connector

TLS Secured Communication **63**

Transmitted Data Encryption **67**

Supporting Incoming IP Address Validation **68**

TLS Secured Communication

For using TLS secured communication for ACF2 Connector, TLS communication must be defined and configured in the following relevant components:

- TCP/IP in the system where ACF2 Server will be active
- Connector Gateway
- SailPoint IdentityIQ or IdentityNow

System Requirements

The following respective components for z/OS versions must be installed for TLS communication:

z/OS Version	Cryptographic Services	z/OS Security Level 3
z/OS 2.5	System SSL Base: FMID HCPT450	System SSL Security Level: FMID JCPT451
z/OS 2.4	System SSL Base: FMID HCPT440	System SSL Security Level: FMID JCPT441

The CSF started task must be active in the system where ACF2 Server is installed.

Implementing Secured Communication for ACF2 Connector

Secured communication to ACF2 Connector must be implemented using **AT-TLS policy**. The TLS processing is done by TCP/IP and is transparent to the ACF2 Connector.

The secured communication is implemented using server authentication.

Implementation Procedure

To set up secure communication for the ACF2 connector:

1. A valid server certificate with its associated server private key must be defined. This certificate must be signed by a trusted Certificate Authority's (CA).

   > **Note**
   > For testing purposes, a local CA can be defined for signing the server certificate.

2. The server certificate and the CA certificate must be connected to a key ring.

3. The CA certificate must be exported to a file, transferred (using FTP with ASCII mode) to the client (with .cer suffix) and installed there to be used for certificate verification by the TLS handshake process.

4. Implement an AT-TLS policy for ACF2 Connector communication.

   > **Note**
   > For detailed information about implementing AT-TLS policy, see "Application Transparent Transport Layer Security data protection" chapter of *z/OS Communications Server IP Configuration Guide*.

   The required policy attributes for AT-TLS policy are:

   - Local Port Range – ports defined in ECAPARM for ACF2 Connector
   - Direction = Inbound
   - TLS Enabled = On
   - TLS v1.1 = On
   - TLS v1.2 = On
   - Handshake Role = Server
   - Client Authorization Type = PassThru
   - Application Controlled = Off
   - Secondary Map = Off

- The name of the certificate created for the secured communication and the name of the key ring to which the server certificate and the CA certificate are connected, should be specified.

> **Note**
> TCP/IP must be granted permission to access the key ring to which the ACF2 server certificate and the CA certificate are connected.

Sample File for AT-TLS Policy

```
# RULE for ACF2 Connector CTSGATE
######################################################
TTLSRule CTSGATE
{
LocalAddr ALL
RemoteAddr ALL
LocalPortRange 2470-2471
Direction Inbound
Priority 255 # highest priority rule
Userid CTSGATE
TTLSGroupActionRef GrpAct_CTSGATE
TTLSEnvironmentActionRef GrpEnv_CTSGATE
TTLSConnectionActionRef GrpCon_CTSGATE
}
TTLSGroupAction GrpAct_CTSGATE
{
TTLSEnabled On
Trace 7
}
TTLSEnvironmentAction GrpEnv_CTSGATE
{
Trace 7
HandshakeRole Server
EnvironmentUserInstance 0
TTLSKeyringParmsRef PrmKeyRing_CTSGATE
TTLSEnvironmentAdvancedParmsRef PrmEnvAdv_CTSGATE
}
TTLSEnvironmentAdvancedParms PrmEnvAdv_CTSGATE
{
TLSv1.1 On
TLSv1.2 On
ClientAuthType PassThru
}
TTLSConnectionAction GrpCon_CTSGATE
{
HandshakeRole Server
TTLSCipherParmsRef PrmCipher_CTSGATE
TTLSConnectionAdvancedParmsRef PrmConAdv_CTSGATE
CtraceClearText Off
Trace 7
}
TTLSConnectionAdvancedParms PrmConAdv_CTSGATE
```

```
{
ApplicationControlled Off
CertificateLabel CTSGATE
SecondaryMap Off
}
TTLSCipherParms PrmCipher_CTSGATE
{
# supported cipher suites - we used a wide list, that should be decreased according
to specific needs
V3CipherSuites TLS_DH_DSS_WITH_DES_CBC_SHA
V3CipherSuites TLS_DH_RSA_WITH_DES_CBC_SHA
V3CipherSuites TLS_NULL_WITH_NULL_NULL
V3CipherSuites TLS_RSA_WITH_NULL_MD5
V3CipherSuites TLS_RSA_WITH_NULL_SHA
V3CipherSuites TLS_RSA_EXPORT_WITH_RC4_40_MD5
V3CipherSuites TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5
V3CipherSuites TLS_RSA_WITH_DES_CBC_SHA
V3CipherSuites TLS_DHE_DSS_WITH_DES_CBC_SHA
V3CipherSuites TLS_DHE_RSA_WITH_DES_CBC_SHA
V3CipherSuites TLS_RSA_WITH_AES_256_CBC_SHA256
V3CipherSuites TLS_RSA_WITH_AES_256_CBC_SHA
V3CipherSuites TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
V3CipherSuites TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
V3CipherSuites TLS_RSA_WITH_AES_128_CBC_SHA256
V3CipherSuites TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256
V3CipherSuites TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256
V3CipherSuites TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
V3CipherSuites TLS_DHE_DSS_WITH_AES_128_CBC_SHA256
V3CipherSuites TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
V3CipherSuites TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
V3CipherSuites TLS_RSA_WITH_AES_128_CBC_SHA
V3CipherSuites TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA
V3CipherSuites TLS_ECDH_RSA_WITH_AES_128_CBC_SHA
V3CipherSuites TLS_DHE_RSA_WITH_AES_128_CBC_SHA
V3CipherSuites TLS_DHE_DSS_WITH_AES_128_CBC_SHA
V3CipherSuites TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
V3CipherSuites TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
V3CipherSuites TLS_RSA_WITH_AES_128_GCM_SHA256
V3CipherSuites TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256
V3CipherSuites TLS_ECDH_RSA_WITH_AES_128_GCM_SHA256
V3CipherSuites TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
V3CipherSuites TLS_DHE_DSS_WITH_AES_128_GCM_SHA256
V3CipherSuites TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA
V3CipherSuites TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA
V3CipherSuites TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA
V3CipherSuites TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA
}
TTLSKeyringParms PrmKeyRing_CTSGATE
{
Keyring CTSRING
}
```

Enabling TLS Between ACF2 Connector and Connector Gateway

For more information on the procedure to be performed on the client side (Connector Gateway), see *SailPoint Integration Guide* or *SailPoint Quick Reference Guide for Gateway Connectors* depending on the Connector Gateway release.

> **Note**
> Ensure that the Transmitted Data Encryption is not active for communication between the ACF2 connector and the Connector Gateway.

Enabling TLS Between SailPoint and Connector Gateway

For more information on the procedure to be performed on SailPoint, refer to the relevant documentation of SailPoint.

> **Note**
> Ensure that the Transmitted Data Encryption is not active for communication between the Connector Gateway and IdentityIQ.

Transmitted Data Encryption

Transmitted Data Encryption must be configured in the ACF2 Connector and in IdentityIQ.

> **Note**
> Transmitted Data Encryption is not supported for communication with IdentityNow.

Implement Transmitted Data Encryption for ACF2 Connector

Install Transmitted Data Encryption Key Dataset

The Connector for ACF2 uses keys in the Encryption Key dataset to encrypt data which is transmitted between Connector for ACF2 and IdentityIQ. The Encryption Key file (global or platform-specific) which was generated in IdentityIQ must be copied to the Encryption Key dataset in Connector for ACF2.

To copy the Encryption Key file, use a file transfer utility available at your site or any other available transfer method. The Encryption Key file is a text dataset. Therefore, the transfer method must convert ASCII to EBCDIC and must support conversion of line breaks.

The Transmitted Data Encryption keys are stored in the following dataset:

```
<prefix>.<version>.ENCREXT
```

where:

- `<prefix>` – Value set for the OLPREFS parameter. For more information, see Connector for ACF2 Datasets Allocation Parameters.
- `<version>` – Value set for the OLVERS parameter. For more information, see Connector for ACF2 Datasets Allocation Parameters.

Implement Transmitted Data Encryption for IdentityIQ

To use Transmitted Data Encryption for IdentityIQ, an Encryption Key file must be created and transferred to the ACF2 Connector, and the Transmitted Data Encryption must be enabled.

For detailed implementation, see *SailPoint Quick Reference Guide for Gateway Connectors* or the *SailPoint Integration Guide*.

Supporting Incoming IP Address Validation

To enable incoming IP address validation, perform the following:

1. In the ECAPARM member of the PARM library, add to the CHANNEL definition: IPLIST=ECAIPLSx statement.
2. A corresponding ECAIPLSx member should be set in the library pointed by DAPARM DD card of the CTSGATE STC.
3. Member ECAIPLSx should contain the relevant Allow and Forbid statements as described in the full feature description that follows.

Configuring Incoming IP Address Validation

Syntax of IP Addresses List

The list of IP addresses is a source table that contains IP addresses of work stations allowed to communicate with CTSGATE that owns this list. An IP address is specified in the standard dotted-decimal notation. From 1 through 4 contiguous from right-to-left sections of an IP address can be specified with an asterisk. Such asterisk in a section of IP address which, actually defines a range of IP address. For example: `80.56.241.*` means in fact, a range of IP addresses from `80.56.241.0` through `80.56.241.255`.

If the above described simplified method of specifying an addresses range turns out too rough to determine a necessary range, then a standard subnet mask can be also used.

Any separate IP address or a range of addresses that should be allowed for communication, must be preceded with the keyword **ALLOW ***. IP address or a range of addresses that should be prevented from communication, must be preceded with the keyword **FORBID ***.

> **Note**
> The first statement of the list must be always [ALLOW *] or [FORBID *].

An incoming address will be checked against the table of IP addresses (in the internal format) to determine if it is to be confirmed or rejected. A given address may in principle match multiple (and even conflicting) entries in the table.

Only one entry will be used to determine whether the IP address is to be allowed or be forbidden. The entry with the most specific address is the effective entry.

The examples provided below demonstrate how a list of IP addresses can be coded.

- The following example allows any IP address, but excluding (forbidding) one specific address (`81.50.1.241`) and a range of addresses (`80.56.241.0 - 80.56.241.255`):

```
ALLOW * First statement allows any IP
Forbid 81.50.1.241
Forbid 80.56.241.*
```

- The same example may be specified with a subnet mask.

```
ALLOW * First statement allows any IP
Forbid 81.50.1.241
Forbid 80.56.241.0, MASK=255.255.255.0
```

- The following example forbids all IP addresses, allowing only two specific addresses (`172.16.241.128` and `81.50.1.241`) and a range of addresses (`80.56.241.0 - 80.56.241.255`):

```
FORBID * First statement forbids any IP
ALLOW 172.16.241.128
ALLOW 81.50.1.241
ALLOW 80.56.241.*
```

- The following example defines 2 short ranges (`172.16.241.0 - 172.16.241.127`) and (`172.16.241.129 - 172.16.241.255`) of addresses that are allowed:

```
FORBID *
ALLOW 172.16.241.*
FORBID 172.16.241.128
```

- The following example shows usage of a subnet mask to allow a range of addresses (`172.16.240.0 -`

172.16.255.255):

```
FORBID *
ALLOW 172.16.240.0, MASK=255.255.240.000
```

- The following is an example of an ALLOW type IP list:

```
FORBID *
ALLOW 172.16.130.151
ALLOW 172.16.110.*
ALLOW 172.16.241.*
ALLOW 80.56.1.*
```

There is no limit on the number of entries in a list of IP addresses. The order of ALLOW and FORBID statements is not important. Source format of an IP list is processed at the initialization of CTSGATE (or when the REFRESH modify command is issued as shown below) to detect syntax errors. Both [ALLOW *] and [FORBID *] statements create the following mask: 0.0.0.0. The absolute value of a mask as a hexadecimal number defines the degree of specificity.

Location of IP Addresses List

The list of IP addresses must reside in a library allocated by a DAPARM DD statement under ECAIPLS fixed name. A one-character suffix is supported in a member name; for example: ECAIPLSx

List Name of IP Addresses List

The name of the required IP list should be specified by the new Channel parameter of CTSGATE as follows:

```
IPLIST=ECAIPLSx
```

The presence of ECAIPLS source member in a library allocated by a DAPARM DD statement is required as soon as the IPLIST channel parameter is specified in ECAPARM.

Modifying the IP Addresses List

ECAIPLS source member is available to a user for changes. ECAIPLSx source member can be refreshed dynamically by means of the modify command REFRESH=ECAIPLSx without the need to restart CTSGATE.

Administration of IP Addresses Validation

- **Operations** – To refresh ECAIPLSx source member dynamically, the following modify command should be issued:

```
F <CTSGATE>, REFRESH=ECAIPLSx
```

> **Note**
> The IPLIST only blocks establishing connections. Refreshing the IPLIST does not affect existing connections.

- **Administrative features** – IP address validation becomes available as soon as:
  - Proper PTF has been applied.
  - A list of IP addresses resides in the library allocated by a DAPARM statement
  - The PLIST channel parameter is specified in ECAPARM.

    TRACE=199 should be set on in order to track processing of the IP list.
- **Security requirements** – The feature is not mandatory. To enable the feature, the IPLIST=ECAIPLSx channel parameter should be specified in the ECAPARM parameters member. When the feature is enabled, a list of IP addresses must exist in a library allocated by a DAPARM statement.
- **Internal diagnostics** – TRACE=199 should be set to ON in order to track processing of the IP list. The ECAIPLSx member will be printed in DAPRENV.

  If the feature is enabled, information about specific IP list will be printed in DAIGLOG output.

  If the CTSGATE channel is disabled due to invalid ECAIPLSx, the detected invalid lines in ECAIPLS will be displayed.

BIND

IP address that IOAGATE must use to listen for incoming connections. If you want IOAGATE to listen on a specific IP address, such as a DVIPA assigned for IOAGATE, use this parameter to identify that IP address.

Use the following syntax:

```
BIND=INADDR_ANY | IP_address | hostname
```

where:

- `INADDR_ANY` instructs IOAGATE to listen for incoming connections from any IP address (adapter) on the system.
- `IP_address` or `hostname` indicates that IOAGATE BINDs to either the given IP_address or the IP_address after hostname resolution.

  Default: `INADDR_ANY`

Operations

The Connector for ACF2 does not include a local user interface since almost all the functions it performs are activated via SailPoint. However, certain Connector for ACF2 operations (generally used for maintenance) are available.

The following topics are discussed in this chapter:

Starting Connector for ACF2 ..... **72**
Shutting Down the Connector for ACF2 ..... **73**
Starting and Stopping the Online Interceptor **73**
Starting the Offline Interceptor Manually **74**
Scheduling the Offline Interceptor **74**
Stopping the Notification and Transaction Servers **77**
Restarting the Notification and Transaction Servers **77**
Viewing System Status **77**

Starting Connector for ACF2

To start the Connector for ACF2:

1. Issue the following operator command:

```
S CTSGATE
```

The Connector for ACF2 Gateway starts and, in turn, automatically starts the Connector servers: Transaction Server (s) and Notification Server. As each of these components of Connector for ACF2 is successfully started, an appropriate message is displayed.

- If the Connector Gateway is active when you start the Connector for ACF2, communication is automatically established between the gateways.
- If the Connector Gateway is not active when you start the Connector for ACF2, the Connector for ACF2 Gateway waits. When the Connector Gateway is started, it establishes communication with the Connector for ACF2 Gateway.

> **Note**
> If Connector for ACF2 will be started using an automatic startup tool during system initialization, the Connector for ACF2 must be started after TCP/IP and the managed ACF2 subsystem have been started.

Shutting Down the Connector for ACF2

To shut down the Connector for ACF2:

1. Issue the following operator command:

```
P CTSGATE
```

The Connector for ACF2 Gateway shuts down the Connector Transaction Server and Notification Server, and then shuts itself down.

Starting and Stopping the Online Interceptor

The Connector Online Interceptor starts separately from the Connector for ACF2 Gateway and the Connector Transaction Server and Notification Server.

> **Note**
> It is recommended that the Online Interceptor be active at all times (even if Connector for ACF2 Gateway is down), in order to ensure that all ACF2 changes are recorded.

Configuring the automated startup of Online Interceptor is described in procedure 9 – Customize Communication Settings.

To start the Online Interceptor:

1. Issue the following operator command:

```
S CTSAONI
```

To shut down the Online Interceptor:

1. Issue the following operator command:

```
P CTSAONI
```

> **Note**
>
> If two or more instances of Connector for ACF2 on the same computer use the dynamic installation of the new password exit, restrictions apply to the order in which the Online Interceptors for each instance of Connector for ACF2 are shut down. The Online Interceptors must be shut down in the reverse order to which they were started.
>
> For example, if the Online Interceptor for Connector instance **A** was started, followed by the Online Interceptor for **B** and then for **C**, the Online Interceptors should be shut down in the order **C**, then **B**, and finally **A**. If two or more instances of Connector for ACF2 on the same computer use the dynamic installation of the new password exit, restrictions apply to the order in which the

> Online Interceptors for each instance of Connector for ACF2 are shut down. The Online Interceptors should be shut down in the reverse order to which they were started.

Starting the Offline Interceptor Manually

To start the Offline Interceptor manually:

1. Issue the following operator command to operate the Offline Interceptor as started task (the procedure is located in the system PROCLIB library):

   ```
   S CTSAOFI
   ```

   The task terminates upon processing of the incremental SMF input.

Scheduling the Offline Interceptor

The Offline Interceptor can be scheduled independently of Connector for ACF2, either manually or automatically via the Notification Server.

The following options can be scheduled for the Offline Interceptor to activate:

- The Notification Server can schedule the Offline Interceptor to run at fixed intervals, starting from the time the Connector for ACF2 is activated. The interval is specified using parameter OFLI_INTERVAL, described below.
- The Notification Server can schedule the Offline Interceptor to run at specific times during the day. The run times are specified using parameter OFLI_RUN_TIME_LIST, described below.
- You can schedule the Offline Interceptor using an external facility.

Configuring Offline Interceptor Parameters

Offline Interceptor scheduling is controlled using parameters in file RSSPARM. This file contains both common and Managed System-specific parameters of the different managed system by Connector for ACF2. Unless indicated otherwise, the parameters (as mentioned below) that control Offline Interceptor scheduling are specified for each Managed System:

- OFLI_INTERCEPT
- OFLI_WAIT_INTERVAL
- OFLI_INTERVAL

- OFLI_RUN_TIME_LIST
- OFLI_RUN_INTERVAL

OFLI_INTERCEPT

Offline Interceptor scheduler activation. Possible values are:

- **Y** – The Offline Interceptor is started periodically by the Notification Server based on the value of parameter OFLI_INTERVAL or OFLI_RUN_TIME_LIST (described below). This is the default value.
- **N** – The Offline Interceptor is not started by the Notification server. You must provide another means of scheduling the Offline Interceptor.

For information on starting the Offline Interceptor manually, see Starting the Offline Interceptor Manually.

OFLI_WAIT_INTERVAL

An interval, starting from the time of the initial communication between Connector for ACF2 and SailPoint, during which the Notification Server will not activate the Offline Interceptor, even if it is scheduled to run during that time. This provides a period at Connector for ACF2 startup during which the administrator can perform functions in Connector for ACF2 without interference from the Offline Interceptor. Format: `hhmmss`. Default: 001000 (10 minutes).

> **Note**
> Unlike the other Offline Interceptor scheduling parameters, OFLI_WAIT_INTERVAL applies to all managed system by the Connector for ACF2 installation.

OFLI_INTERVAL

The interval that must pass between consecutive activations of the Offline Interceptor (format: `hhmmss`). The Offline Interceptor is first activated at the earliest opportunity following the activation of Connector for ACF2, and then after the specified interval. If Connector for ACF2 is stopped and restarted, the Offline Interceptor is not activated until the specified interval has passed from the previous activation. Default: 010000 (1 hour).

OFLI_RUN_TIME_LIST

A list of times at which the Offline Interceptor should be activated. The times are stated in 24-hour format `hh:mm`, separated by commas. `00:00` represents midnight.

For example:

```
00:00,03:00,09:30,12:00,15:30,21:25
```

The Offline Interceptor will be activated at the times specified in the list. If, for any reason (for example, Connector for ACF2 was not active), the Offline Interceptor missed a scheduled activation, it will not be activated until the next time specified in the list. See also OFLI_RUN_INTERVAL parameter.

If OFLI_RUN_TIME_LIST and OFLI_INTERVAL parameters are specified, Connector for ACF2 ignores OFLI_RUN_TIME_LIST parameter.

If neither of the parameters are specified, Connector for ACF2 assigns the default value **02:30** to the OFLI_RUN_TIME_LIST parameter.

OFLI_RUN_INTERVAL

The maximum deviation from scheduled activation time (in parameter OFLI_RUN_TIME_LIST) that the Notification Server will tolerate for activation of the Offline Interceptor. If the Offline Interceptor cannot be activated within this interval from the scheduled time (for example, Connector for ACF2 was not active), the scheduled activation is skipped. Format: `hhmmss`. Default: `001000` (10 minutes).

> **Note**
> Parameters OFLI_INTERVAL, OFLI_WAIT_INTERVAL and OFLI_RUN_INTERVAL must be expressed as valid time representations in the format `hhmmss`, where `hh` is 00 through 23, `mm` is 00 through 59, and `ss` is 00 through 59. For example: `010000` translates to 01:00:00 (1 hour) and is a valid value; 006000 translates to 00:60:00 which is not a valid time and thus will be ignored.

Example:

Given the following parameter values:

- OFLI_RUN_TIME_LIST : `00:00,12:00`
- OFLI_RUN_INTERVAL: `020000` (2 hours)
- OFLI_WAIT_INTERVAL: `001000` (10 minutes)

Scheduling of the Offline Interceptor by the Notification Server would be performed as follows:

- If Connector for ACF2 was shut down at `23:00` and then restarted at `01:30`, the Offline Interceptor would be activated at `01:40` (Connector for ACF2 start time + OFLI_WAIT_INTERVAL), as this is within the 2-hour deviation permitted by parameter OFLI_RUN_INTERVAL.
- If Connector for ACF2 was shut down at `23:00` and then restarted at `01:55`, the Offline Interceptor would not be activated for its scheduled run time of 00:00. This is because the earliest possible activation time for the Offline Interceptor is `2:05` (Connector for ACF2 start time + OFLI_WAIT_INTERVAL). This exceeds the 2-hour deviation permitted by parameter OFLI_RUN_INTERVAL by 5 minutes. The next activation of the Offline Interceptor would occur at `12:00`.

Stopping the Notification and Transaction Servers

The Notification Server and/or Transaction Server can be stopped without having to shut down the Connector for ACF2 Gateway.

To stop the Notification Server or Transaction Server:

1. Issue the following command:

   ```
   F CTSGATE,STOPASID=<nn>
   ```

   where <*nn*> is the ID of the server to be stopped.

Example:

The Notification Server is typically server 01 and the Transaction Server is server 02. Given this situation, you would use the commands that follow.

- To stop the Notification Server: `F CTSGATE,STOPASID=01`
- To stop the Transaction Server: `F CTSGATE,STOPASID=02`

Restarting the Notification and Transaction Servers

If either the Notification Server or a Transaction Server terminates for any reason, it can be restarted without having to shut down the Connector for ACF2 Gateway.

To restart the Notification Server or Transaction Server:

1. Issue the following command:

   ```
   F CTSGATE,STARTASID=<nn>
   ```

   where <*nn*> is the ID of the server to be restarted.

Example:

The Notification Server is typically server 01 and the Transaction Server is server 02, 03 or 04. Given this situation, you would use the commands that follow.

- To restart the Notification Server: `F CTSGATE,STARTASID=01`
- To restart a Transaction Server: `F CTSGATE,STARTASID=02, 03 or 04`

Viewing System Status

Display the list of servers and their statuses. If the server is currently handling a specific service, the service is also displayed.

To view the system status:

1. Issue the following command:

```
F CTSGATE,STATUS
```

Connector for ACF2 Implementation

The Connector provides the means by which SailPoint, the ACF2 Connector, and ACF2 can "understand" one another.

The ACF2 Connector presents a uniform interface on all platforms. The Connector is designed to interact with ACF2 according to its specific characteristics and capabilities. Each Connector for ACF2 function is customized for ACF2 and enables the interception of Managed System events and the translation of SailPoint commands to ACF2 terminology.

The following topics are discussed in this chapter:

Multi-Valued Logonid Fields **79**

ACF2 Group Representation **80**

Implementing Groups **84**

Connector and ACF2 Interaction **88**

Multi-Valued Logonid Fields

By including a multi-valued field in the UID structure, it is possible to create multiple views of the UID for a single user.

The following example demonstrates possible implementation of this feature.

Given that the UID structure consists of the following fields:

Parameter	Description
ORG (organization)	1 character
DPT (department)	2 characters
PRJ (project)	4 characters (Multi-valued field)
LID	8 characters

User fields are assigned the following values:

- **ORG (D)**
- **DPT (AP)**
- **PRJ(Y2KP,ACCP,SECU)** (This field is assigned multiple values.)
- **LID(US908622)**

As a result, the user has the following UID strings:

- **DAPY2KPUS908622** (primary UID)
- **DAPACCPUS908622**
- **DAPSECUUS908622**

> **Note**
> Except for the multi-valued field, which has a different value in each UID view, all fields have the same value in all UID views.

When a user attempts to access a resource, all the user's UID views are checked for authorization. If at least one UID string matches, access is permitted.

Supporting Multi-Valued Logonid Fields

SailPoint fully supports multi-valued Logonid fields and multiple views of the UID for a single user.

The following features are included in SailPoint as part of multi-valued Logonid field support:

- Ability to insert, update, delete and retrieve any or all values from a multi-valued logonid field.
- A user's primary UID (or the user's only UID if the user has a single UID string) is displayed in field **UID** of the Account Properties window in SailPoint.
- If a user has more than one UID string, all UID string views will be displayed in field **Secondary UIDs** of the Account Properties window in SailPoint.

ACF2 Group Representation

Most Managed System have a built-in user group entity. This entity is used to give multiple users common access authorities. When a group is defined and connected to users, all the users connected to a group have access to all the resources permitted to that group. By allowing a user to be connected to many groups at the same time, groups can be created to represent organizational structure, applications, etc.

ACF2 has no group entity. The administrator grants access rights for a given resource to multiple users by using a UID string mask in the access rule. Access rule permissions apply to all the users matching the UID mask specified in the rule. UID mask can be used to represent a group of users, much like group objects in other Managed System.

Controlling a user's UID string is a tedious and error-prone task since, in many cases, the UID is made up of fields and the information within it is coded.

To simplify administration of ACF2, SailPoint provides the functional equivalent of user groups via the Group entity. Group entities are defined and managed using SailPoint.

A group defined in SailPoint has a name and description and is associated with ACF2 UID masks. Group definitions are stored in Connector for ACF2 controlled dataset.

Benefits of Group Implementation

The advantages offered by the implementation of groups in SailPoint are:

- Logical, meaningful names can be used for UID masks, reflecting their grouping qualities.
- Setting up a new user is easily accomplished by connecting the user to groups or disconnecting the user from groups instead of updating the coded UID fields.
- The group can be associated with SailPoint IT Roles, allowing for automated creation of users and assigning of access rights.
- The Account—Group connection window shows which Account is associated with a Group and vice-versa.
- SailPoint's drag-and-drop capabilities can be used to create connections between Accounts and Groups.

Group Entity

Each group defined in SailPoint has a UID mask associated with it. A user is considered to be connected to a group if the user’s UID string matches the UID mask associated with the group.

Example

Given the following situation:

- Each user’s UID string is made of the fields AP1, AP2 and LID where AP1 and AP2 are single character fields representing access to applications A, B and C.
- A group named "Group A" is associated with UID mask: A**********
- User John has the following UID string: A JOHN (AP1=A, AP2=blank, LID=JOHN).

Since John’s UID string matches "Group A" UID mask, John is considered to be connected to "Group A".

Connecting Users to Groups

When a user is connected to a group, the non-asterisk characters of the UID mask associated with the group are imposed on the user's UID fields, thus enabling the user access to resources permitted to the UID mask associated with the group.

Disconnecting the user from the group "blanks out" the characters that "connect" the user to the group, thus disabling the user's access to the resource permitted to the group.

Example 1

Given the following situation:

- Each user's UID string is made of the fields AP1, AP2 and LID where AP1 and AP2 are single character fields representing access to applications A, B and C.
- User John has the following UID string: A JOHN (AP1=A, AP2=blank, LID=JOHN), which means John can access application A.
- A new group named "Group B" is defined which is associated with UID mask: *B*********

By connecting John to "Group B", John's UID string is set to ABJOHN, thus allowing John to access Application B.

By next disconnecting John from "Group B", John's UID string is reset to A JOHN and removes John's authority to access Application B.

> **Note**
> A user can only be connected to a group if the user's UID string has a blank character in the field required by the group's UID mask.

Example 2

Given the following situation:

- Each user's UID string is made of the fields AP1, AP2 and LID where AP1 and AP2 are single character fields representing access to applications A, B and C.
- Group "Group B" is defined with UID mask: *B*********
- User John is connected to "Group B".
- A new group name "Group C" is associated with UID mask: *C*********

John cannot be connected to "Group C", since the second character (field AP2) of his UID string already contains the value B. This usage conflicts with the UID mask of Group C.

To connect John to "Group C", John must first be disconnected from "Group B", thus clearing the second character of his UID string.

Defining Groups with Multiple UID Masks

There are cases where different UID masks represent the same meaning in the access rules. For example, an access rule may state that a user has access to application A if the user's UID string contains an "A" in either the first or second position.

SailPoint implementation allows the association of multiple logically-equivalent UID masks with the same group.

Example 1

Given the following situation:

- Each user's UID string is made of the fields AP1, AP2 and LID where AP1 and AP2 are single character fields representing access to applications A, B and C.
- ACF2 access rules are set up so that if a user has an "A" either in AP1 or AP2, the user has access to Application A.

This situation can be defined using either of the following configurations:

- Group "Appl A1", associated with UID mask A*********
- Group "Appl A2", associated with UID mask *A********

The administrator can grant a user access to Application A by connecting the user to either group.

- One group ("Appl A") can be defined, associated with UID masks A******** and *A*******.

The administrator can grant a user access to Application A by connecting the user to group "Appl A".

In cases where a group definition contains several alternate masks, a user is considered to be connected to the group if it matches at least one of the UID masks.

> **Note**
>
> When connecting a user to a group with multiple UID masks, Connector for ACF2 applies the first mask which does not conflict with the user's existing UID string.
>
> When disconnecting a user from a group, SailPoint clears all the UID string characters matching any of the group UID masks.

Example 2

Given the following situation:

- Group "Appl A" is associated with UID masks A******** and *A*******.
- User John has the following UID string: B JOHN (AP1=B, AP2=space, LID=JOHN).
- The administrator connects user John to group "Appl A."

The first UID mask in group "Appl A" cannot be applied to John's UID string since AP1 is already set to allows access to Application B. However, the second UID mask in the group (in which AP2=A) does not conflict.

As a result, the connect operation succeeds using the second mask. John's UID string after the operation will be BAJOHN.

Multi-Valued Logonid Field Considerations

When a user's UID string contains a multi-valued field, the behavior of connect and disconnect operations differ from the ones described above.

Unlike regular fields, all characters belonging to a multi-valued field are considered to be one block, which can be either fully masked or contain a full value. A group UID mask cannot contain a partial mask on a multi-valued field.

The topics which follow describe how connect and disconnect operations are implemented when the user's UID field contains a multi-valued field.

Connect Operations

If the UID mask of the group to connect requires adding a new value to the user's UID string in a multi-valued field, one of the following occurs:

- The new value is inserted into a field in the user's UID string. The user's UID string does not contain any value in that field. The value is inserted and the connection is successful.
- A new value is added to the multi-valued field in the user's UID string. In a single-value field, if a conflict exists between the user's UID string and the group's UID mask, the connect operation is impossible since the field is already occupied. However, in a multi-valued field, it is possible to add a new value to the list of existing values for that field. If the number of existing values does not exceed the maximum number allowed, the connect operation succeeds. Otherwise, the connect operation fails.
- No change occurs in the user's UID string. If the user's UID string (or UID string views) already contains the value designated in the UID mask, the connection already exists.

Disconnect Operations

If the group's UID mask forces a value to be removed from a multi-valued field, one of the following occurs:

- A value is deleted from the list of existing values for the field. In a multi-valued field, it is possible to delete a value from the list of existing values for that field, causing the removal of one UID string view.
- The field is assigned blank characters. If a multi-valued field contains only one remaining value, the value is assigned blank characters rather than being deleted. This is similar to the treatment of a single-value field.
- The user's UID string (or UID string views) does not contain the value designated in the UID mask. In this case, the connection does not exist.

Implementing Groups

The group entity is a security administration tool which simplifies ACF2 user administration. It does not modify the manner in which ACF2 normally operates. The group labels a set of one or more UID masks that are used later in

connect/disconnect User-User Group operations.

Implementation of groups is optional. You may choose to directly update the user's UID fields as it is done in ACF2 instead of using the Group entity in IdentityIQ. The implementation of groups may be done in parallel with regular implementation of Connector for ACF2.

Defining Groups and Selecting UID Masks

A group is defined via SailPoint. After the group is defined, you can use the IdentityIQ to view all users connected to the group definition (that is, users that match any of the group's UID masks). These users are implicitly connected to the group without the execution of a connect operation.

Each group may represent a single user attribute reflected in the user's UID string (Division, Department, Business Role, Application). By connecting a user to a group, the LID record attribute is set.

The attribute associated with a group may contain several UID components (for example, Department and Business Function). The decision to create a single group with multiple attributes or multiple groups with a single attribute depends on site requirements.

Example 1

UID structure is: DEPT, ROLE, LID

You can define a group for each ROLE or DEPT value or for every combination of values for these attributes.

- UID string for user JOHN is `FINMGRJOHN (DEPT=FIN; ROLE=MGR)`
- UID string for user FRED is `PAYACCFRED (DEPT=PAY; ROLE=ACC)`

If each user must have both DEPT and ROLE (and no other situation is allowed), the following two groups may be defined:

Group	UID mask
Finance Managers	FINMGR********
Payroll Accountants	PAYACC********

JOHN is considered connected to group "Finance Manager" and FRED is considered connected to group "Payroll Accountant."

Example 2

Using the same UID structure as in the previous example, every user may be assigned either DEPT or ROLE, or both.

- UID string for JOHN is: `FIN...JOHN (DEPT=FIN; ROLE= space)`
- UID string for FRED is: `...ACCFRED (DEPT= space; ROLE=ACC)`

- UID string for GREG is: `FINACCGREG (DEPT=FIN; ROLE=ACC)`

Accordingly, the following groups are created:

Group	UID mask
Finance Department	FIN**********
Managers	***MGR********
Payroll Department	PAY**********
Accountants	***ACC********

Users FRED, JOHN and GREG are connected to the groups as follows:

- JOHN is connected to group FINANCE Department.
- FRED is connected to group Accountants.
- GREG is connected to both Finance Department and Accountants.

SailPoint Connector for ACF2 does not allow the definition of two groups which overlap. In a case where two groups overlap, a user connected to one group is automatically connected to the other group.

Example 3

The UID structure is: DEPT, ROLE, LID

The UID string for user JOHN is: `FIN...JOHN`

Two groups are defined:

Group	UID mask
Finance Department	FIN**********
Finance Manager	FINMGR********

JOHN is currently associated with group Finance Department.

If JOHN is connected to the Finance Manager group, his UID string becomes: `FINMGRJOHN`

> **Note**
> Any attempt to disconnect either group from JOHN disconnects the other group as well. If JOHN is associated with these two groups via IT Roles, one of the IT Roles is invalidated via the disconnect operation of another IT Role.
>
> To avoid these issues, define two separate groups for the Department and Role and allocate them selectively.

Example 4

Some users may have several roles under a given department. In this case, a multi-valued field can be used to assign multiple roles to a single user.

The UID structure is: DEPT, ROLE, LID, where ROLE is a multi-valued field.

The UID string views for user JOHN are:

- `APPDVLJOHN (DEPT=APP; ROLE=DVL)`
- `APPDBAJOHN (DEPT=APP; ROLE=DBA)`

User JOHN, who is a developer in the Applications department, is also the database administrator (DBA) for that department.

The following groups are defined:

Group	UID mask
Developers	***DVL********
Database Administrators	***DBA********

JOHN is considered connected to both groups.

Conclusion

Before implementing groups, consider the following:

- The various values used in the site's UID attributes for users.
- The manner in which UID attribute values are assigned to users.

The approach of "Group = Single Attribute" ensures that no group conflicts are created. On the other hand, creating more specific groups eases system administration.

Updating UID Masks for a Group

The UID masks are the essence of groups. By changing a group's UID masks, the set of users connected to the group changes instantly.

To avoid administration inconsistency, the group's UID masks may be changed only when the group is not associated with IT Roles. If a group is changed while part of a IT Role, a discrepancy may occur as users will be indirectly dis-connected from the group. To avoid this, IdentityIQ prevents the administrator from changing UID masks for a group if the group is a member of an IT Role.

Connector and ACF2 Interaction

The Connector is designed to interact with ACF2 and update or retrieve information from it. There are two types of ACF2 actions that are controlled by the Connector functions.

Transactions which originate in SailPoint include:

- User definition
- User password change
- User group definition
- User to user group connection
- Resource Access Control List (ACL) definition as part of targetAggregation or as part of Account and Group aggregation processes

Platform Based Operations

The Connector is designed to be called for events and alerts that occur in ACF2 which do not originate in SailPoint. These operations include:

- User definition
- User password or phrase change

The Connector Interceptors are responsible for calling the relevant Connector for ACF2 function for the above events. The Connector for ACF2 is responsible for propagating the events to SailPoint.

Scripts

The Connector for ACF2 functions (described in this book) are designed to meet fundamental Managed System requirements.

In addition to the basic requirements, site-specific requirements may exist in established Managed System, and Managed System may have changing needs. Therefore, Connector for ACF2 enables customization of the functions. You can customize any Connector for ACF2 function by writing scripts.

A script is a group of statements that perform one or more actions and that manipulate standard SailPoint and Managed System-specific fields. A script may include conditions that determine when actions should be performed. For example, cleanup activities should be run if the Connector for ACF2 function does not execute successfully.

Scripts can also be used to automate any required actions that are currently performed manually.

The Connector for ACF2 has the ability to call a script before and after executing a Connector for ACF2 function. Parameters may be received from and returned to Connector for ACF2 functions by including script commands in the scripts.

Writing a Script **89**

Executing a Script **91**

Script Variables **92**

Setting the Return Code **98**

TSO Considerations **98**

Script Commands **99**

Writing a Script

Scripts are written in REXX language and must adhere to REXX syntax rules. Use a text editor to write the scripts and store them in the following library:

```
<prefix>.<version>.USER.CLIST
```

where:

- `<prefix>` – Value set in parameter OLPREFS in member LOADCTS in the Connector INSTALL library.
- `<version>` – Value set in parameter OLVERS in member LOADCTS in the Connector INSTALL library.

> **Note**
> This library name is defined in parameter SCRIPT_DIR in member RSSPARM stored in the Connector PARM library.

Scripts may be executed before and/or after a Connector for ACF2 function is executed (see Executing a Script for details). All scripts are activated under the Connector for ACF2 address space.

A script should consist of the following sequence of actions:

1. Read the current Connector for ACF2 variables into REXX variables.
2. Examine the REXX variables, modify them (if required) and execute additional actions as required.
3. Update the Connector for ACF2 variables with the resultant REXX variable values.

To enable the script to manipulate Connector for ACF2 variables, Connector for ACF2 provides the script command CTSAVAR which reads the current Connector for ACF2 variables into REXX variables. After the REXX variables have been examined and modified, the script command CTSAVAR is used to update the Connector for ACF2 variables with the resultant REXX variable values. Script commands are described later in this chapter.

Since the script is executed in the TSO-REXX environment, Connector for ACF2 scripts can also issue TSO and REXX commands.

When writing a script, the following must be taken into account:

- Certain variables are unmodifiable. For more information, see Script Variables.
- The return code of a script must be set by the script before it terminates for more information see Setting the Return Code.
- Several TSO commands can only be issued indirectly from within a Connector for ACF2 script. For more information, see TSO Considerations.
- Due to REXX limitations, field names for keywords must be all uppercase. This applies both when defining keywords in SailPoint and when specifying the field name of the keyword in a script.
- The simplest way to return a message from a script to SailPoint is by issuing a SAY REXX command with the relevant text.

  For example:

```
SAY "THIS IS AN ERROR MESSAGE…"
```

> **Note**
> For SailPoint to display this message, the transaction should complete with a non-zero return code.

Executing a Script

The Connector for ACF2 determines which scripts to execute and whether or not to run the scripts by examining the values of the following parameters in member RSSAPI in the Connector PARM library.

Pre/Post-Scripts

One script can be called before a Connector for ACF2 function executes and another script (or the same one) can be called after the Connector for ACF2 function executes. The scripts are referred to as:

- **Pre-script** – This script is run immediately before the Connector for ACF2 function executes.
- **Post-script** – This script is run immediately after the Connector for ACF2 function executes.

The Connector for ACF2 execution of the functions is as follows:

1. If a pre-script is specified and should be executed, the Connector for ACF2 invokes it. Otherwise, execution of the Connector for ACF2 function begins with step 3.
2. The return code of the pre-script is checked. If the script returns **SKIP** or **FATAL**, the Connector for ACF2 does not invoke the functions and execution skips to step 4.
3. If the Connector for ACF2 function is to be invoked, it is called.
4. If a post-script is specified and should be executed, the Connector for ACF2 invokes it, regardless of the return codes from the pre-script and the actual Connector for ACF2 function.
5. The final return code is the return code of the last item (Pre/Post-script or actual Connector for ACF2) that was executed.

If both of the following conditions are satisfied:

- The Connector function is an ADD or UPDATE function.
- A GET post-script is specified for the object and should be executed.

  Connector for ACF2 invokes the GET post-script after invoking any ADD or UPDATE Pre/Post-scripts required by the function. This is done to synchronize the Managed System database with the SailPoint database.

Structure of the RSSAPI Member

Each line in the RSSAPI member (excluding comment lines) represents a Connector for ACF2 call which is followed by various parameters. The parameters influence the behavior of the Connector for ACF2. Each line conforms to the following format:

```
Managed System-type Managed System-name API-name NUM PRF ACF POF PRN PON
```

The following table describes the parameters appearing in each line of the RSSAPI member:

Parameter	Description
Managed System-type	Managed System type
Managed System-name	Managed System name
API-name	Connector for ACF2 functional name.
NUM	Maximum number of keywords that may be added to a transaction by the Pre-script for use by the Connector for ACF2 function or Post-script.
PRF	Y/N flag indicating whether or not to invoke the Pre-script.
ACF	Y/N flag indicating whether or not to invoke the actual Connector for ACF2 routine.
POF	Y/N flag indicating whether or not to invoke the Post-script.
PRN	Pre-script name.
PON	Post-script name.

Example:

`- MVS ADDUSER 05 N Y Y ACFADDU1 ACFADDU2` – This line indicates that the Connector for ACF2 function ADDUSER in Managed System MVS operates as follows:

- Does not call the Pre-script ACFADDU1.
- Performs the actual Connector for ACF2 routine.
- Calls the post-script ACFADDU2.

A script can be called as a pre-script and/or post-script for any Connector for ACF2 function. For example, assume that you wish to perform the same action before every Connector for ACF2 function. First write a script which performs the desired action. Next, specify the script name in parameter *PRN* for all the Connector for ACF2 functions and set the parameter *PRF* to Y.

Script Variables

When executed within the Connector for ACF2 environment, a script has access to certain predefined variables. These variables are used for a variety of purposes, including:

- Receiving and modifying values of Managed System-specific fields
- Receiving information regarding the Connector for ACF2 environment
- Controlling actions performed by the Connector for ACF2 function
- Controlling execution of subsequent scripts

These topics are described in detail in this section and following sections.

Types of Variables

The Connector for ACF2 distinguishes between the different kinds of information that is passed to and from the scripts by using different categories of variables; each category of variable is assigned a different prefix.

The categories of variables are:

- **CTSA0.<field_name> (Read-only)** – Predefined variables representing SailPoint structure fields (common to all Managed System types) or Connector for ACF2 environment information.

  For a full list of CTSA0 variables, see CTSA0 Variables.

- **CTSA1.<field_name> (Read/write)** – Predefined Managed System-specific variables containing values sent from SailPoint. If the value of a field was modified in SailPoint, the corresponding CTSA1 variable contains the modified value. The value of the CTSA1 variable can be further modified by the script.

  For more information, see Appendix E: Managed System-Specific fields.

  In addition to the variables described above, the following special CTSA1 variables are available:

  - **CTSA1.MSG (Write-only)** – Used to send a message from the script to SailPoint. This variable is defined by the script; use of the variable is optional. A string value assigned to this variable is sent to SailPoint and is displayed in the Transaction Properties window.
  - **CTSA1.RC (Write-only)** – Return code of the pre- or post-script. Before Connector for ACF2 executes a function, it checks the pre-script return code to determine whether or not to execute the function. Therefore, the return code of a pre-script must be set by the script before it terminates.

    For more information, see Setting the Return Code.

- **CTSA2.<field_name> (Read-only)** – Predefined Managed System-specific variables containing the original values of fields. The values are retrieved from the Managed System.

  These variables are similar to CTSA1 variables. The same field can be referenced with both the CTSA1 and CTSA2 prefixes. However, the CTSA1 variable contains the new value of the field as sent from SailPoint while the CTSA2 variable contains the original "old" field value.

After the Managed System has been updated, each CTSA2 variable passed to the script contains the new, updated value of the corresponding field.

- **CTSA9.<rss_parameter> (Read-only)** – Variables containing values of parameters in the RSSPARM file.

  The script only receives CTSA9 variable values when the parameter SEND_RSSPRM_TO_SCRIPT in the RSSPARM file is assigned the value Y.

List Fields

Managed System-specific information may include List (table) fields. A List field is passed to a script as a string of entries. This string includes specific characters that are designated to separate entries and sub-fields in the list.

The default entry separator value is a comma (,). To specify a value other than the default, assign any printable character to the Managed System parameter SCRIPT_ SEP_ENTRY in the RSSPARM file.

The default sub-field separator value is a semicolon (;). To specify a value other than the default, assign any printable character to the Managed System parameter in SCRIPT_SEP_FIELD in the RSSPARM file.

For example, given the following:

- SCRIPT_ SEP_ENTRY is set to ^
- SCRIPT_SEP_FIELD is set to $

A List field contains the following data:

A1	A2	A3
B1	B2	B3
C1	C2	C3
D1	D2	D3

The above table of data values will be passed to the variable in a script as:

```
A1$A2$A3^B1$B2$B3^C1$C2$C3^D1$D2$D3
```

CTSA0 Variables

CTSA0 variables and their values are in the following topics. Note that the variables are listed according to Connector for ACF2 function type.

Connector for ACF2 Environment Variables

The following table lists variables containing information that relates to the environment in which the script is executed. These variables are common to all Connector for ACF2 function types.

Variable	Value	Description
CTSA0.ACTION	SCRIPTPRE	Call as a Pre-script
	SCRIPTPOST	Call as a Post-script
CTSA0.FUNC_NAME	ADDU2UG	Connect a user to a group
	ADDUG	Create a new group
	ADDUSER	Create a new user
	DELU2UG	Disconnect a user from a group
	DELUG	Delete a group
	DELUSER	Delete a user
	GETUGS	Obtain group details
	GTRSPRM	Obtain Managed System details
	GTUG2UC	Obtain user to group connection details
	GTUSERS	Obtain user details
	REVUSER	Revoke/Restore a user
	UPD_PASS	Update password of a user
	UPDUSER	Update user details. **Note** When a transaction is issued from SailPoint for any of the following actions: • changing the user's password • revoking the user • restoring the user With no added Managed System-specific or user-defined fields, the transaction type can be either UPDUSER, or it can be UPD_PASS / REVUSER, depending on how the action was initiated. However, when the transaction includes added Managed System-specific or user-specific fields, the transaction type is always UPDUSER.
	UPDU2UG	Update user to group connection details
	UPDUG	Update group details
CTSA0.ACT_RC	OK	Return code of the actual Connector for ACF2. Available in

Variable	Value	Description
		the Post-script only
	ERROR	
	FATAL	
	<UNDEFINED>	
CTSA0.ADM_G		Group of administrator performing the operation
CTSA0.ADM_VER	4.0.01	Current version of Connector for ACF2.
CTSA0.ADM_MOD	1	Reserved.
CTSA0.ADM_ID		User ID of administrator performing the operation.
CTSA0.ADM_PASSWD		Managed System Administrator password or phrase. This parameter is passed to the scripts, only when the parameter SEND_PWD_TO_SCRIPT in the file RSSPARM is set to Y.
CTSA0.PRE_RC	OK	Return code of the Pre-script. Available in the Post-script only
	WARN	
	SKIP	
	ERROR	
	FATAL	
	<UNDEFINED>	

Managed System User Connector for ACF2 Function Variables

The following table describes variables available for any Managed System user operation.

Variable	Value	Description
CTSA0.USER_ID		User ID
CTSA0.USER_PWD	<password or phrase>	Managed System user new password or phrase. This parameter is passed only when parameter PASS_PASSWORD in file RSSPARM is set to Y.
CTSA0.UG_DEF		Default group of the user
CTSA0.USER_ADMIN	1	User is a regular user
	2	User is an auditor
	3	User is an administrator
	4	User is an auditor and administrator
	5	Ignore this field

Variable	Value	Description
CTSA0.USER_STA	1	User is revoked
	2	User is restored
	3	Ignore this field
CTSA0.PWD_LIFE	1	Permanent
	2	Temporary
	3	Ignore this field
CTSA0.RSS_NAME		Name of the Managed System
CTSA0.RSS_TYPE		Type of Managed System

Group Connector for ACF2 Functions

The following table describes variables available for any Group Connector for ACF2 operation.

Variables	Value	Description
CTSA0.GROUP_ID		Group ID
CTSA0.GROUP_PR		Parent Group
CTSA0.RSS_NAME		Name of the Managed System
CTSA0.RSS_TYPE		Type of Managed System

Managed System User–Group Connector for ACF2 Functions

The following table describes variables available for any Managed System User—Group Connector for ACF2 operation.

Variable	Value	Description
CTSA0.GROUP_ID		Group ID
CTSA0.USER_ID		User ID
CTSA0.U2UG_ATR	1	Regular connection between a user and a group
	2	Connection is of user to its default group
	3	Ignore this field
CTSA0.U2UG_MSC	1	User is a regular member of the group
	2	User is an administrator of the group
	3	User is an auditor of the group
	4	User is administrator and auditor of the group
	5	Ignore this field
CTSA0.RSS_NAME		Name of the Managed System
CTSA0.RSS_TYPE		Type of Managed System

Setting the Return Code

Before Connector for ACF2 executes a Connector function, it checks the Pre-script's return code to determine whether or not to execute the function. Therefore, the return code of a Pre-script must be set by the script before it terminates.

To set the return code in a Pre-script:

1. In the script, set variable **CTSA1.RCODE** to one of the following values:

Value	Description
OK	Pre-script processing completed successfully.
SKIP	Do not call the Connector for ACF2 function, but do call the Post-script.
WARN	Continue and call both the Connector for ACF2 function and the Post-script.
ERROR	Continue and call both the Connector for ACF2 function and the Post-script.
FATAL	Do not call the Connector for ACF2 function, but do call the Post-script.

To set the return code in a Post-script:

1. In the script, set variable **CTSA1.RCODE** to one of the following values:

Value	Description
OK	Post-script processing completed successfully.
WARN	Post-script processing completed with warning.
SKIP	(GET Post-script only) Do not send the retrieved object (for example, Managed System user, resource) to SailPoint. Using this value enables a Post-script to filter the objects to be aggregated to SailPoint, regardless of the aggregation.
ERROR	Post-script processing completed with error.
FATAL	Post-script processing completed with fatal error.

TSO Considerations

The TSO environment is available to scripts and therefore most TSO commands can be issued by the script. However, the following TSO commands *cannot* be issued directly from within a Connector for ACF2 script.

- Authorized TSO commands
- SUBMIT command

These commands and the method in which they can be issued are described below.

Authorized TSO Command

Authorized TSO commands cannot be issued directly from a script.

To execute an authorized TSO command:

1. Use the CTSAEXC command processor.

   For example, to activate command IDCAMS DEFINE (which is an authorized command processor) enter the following statement in the script:

   ```
   CTSAEXC DEFINE ....define command arguments...
   ```

SUBMIT Command

The TSO SUBMIT command cannot be issued directly from a script.

To submit a job:

1. Use the CTSASUB command processor. CTSASUB receives the name of a DD statement which contains the job JCL image.

   For example, include the following statement in the script:

   ```
   CTSASUB TEST
   ```

   This activates command CTSASUB which reads DD statement TEST and writes its contents to the JES internal reader. An exclusive JES internal reader is allocated to the Connector for ACF2 started task via DD statement INTRDR in the task's JCL. If no DD statement is specified for command CTSASUB, a default DD statement CTSJOBIN is used and its contents are written to the JES internal reader.

Script Commands

Thefollowing table provides a brief description of the available script commands.

Command	Description
CTSAEXC	Activates authorized TSO command processors.
CTSASUB	Submits a job.

Command	Description
CTSAVAR	Copies Connector for ACF2 variables to REXX variables. *OR* Updates Connector for ACF2 variables based on the REXX variables.
CTSASYNC	Generates a synchronization event.

CTSAEXC

Purpose	Activates authorized TSO command processors in the Connector for ACF2 script
Syntax	`CTSAEXC <TSO_command>` where `<TSO_command>` is any authorized TSO command.
Description	Activates authorized TSO command processors in a Connector for ACF2 script.
Example	The following statement activates the IDCAMS DEFINE command: `CTSAEXC DEFINE ALIAS NAME(USER1) RELATE (CATALOG.USERS)`

CTSASUB

Purpose	Submits jobs from a Connector for ACF2 script
Syntax	`CTSASUB [<ddname>]` where `<ddname>` is the name of the DD statement which contains the JCL cards.
Description	Submits jobs from a Connector for ACF2 script.
Example	The following statement submits the job contained in the dataset allocated to DD statement MYJOB: `CTSASUB MYJOB`

CTSAVAR

Purpose	Copies Connector for ACF2 variables to REXX variables or updates Connector for ACF2 variables based on the REXX variables
Syntax	`CTSAVAR {GET | PUT} <token>` where: • `GET` – Sets the REXX variables to the values in the Connector variables. • `PUT` – Updates the Connector variables to the values in the REXX variables. • `<token>` – Token passed to the script as a parameter which is used to identify the relevant Connector for ACF2 parameters by the CTSAVAR command.
Description	Copies Connector for ACF2 variables to REXX variables or updates Connector for ACF2 variables based on the REXX variables.
Example	The following statement sets the REXX variables to the values in the Connector for ACF2 variables: `CTSAVAR GET <token>`
	The following statement updates the Connector for ACF2 variables to the values in the REXX variables: `CTSAVAR PUT <token>`

CTSASYNC

Purpose	Triggers synchronization of Managed System details with SailPoint
Syntax	One of the following: • `CTSASYNC <token> USER <user>` • `CTSASYNC <token> GROUP <group>` • `CTSASYNC <token> CONN <conn_user>`

Purpose	Triggers synchronization of Managed System details with SailPoint
	`<conn_group>` where: • `<token>` – Token passed to the script as a parameter which is used to identify the relevant Connector parameters by the CTSASYNC command. • `<user>` – User name of the user to be synchronized. • `<group>` – Group name of the group to be synchronized. • `<container>` – Container name of the container to be synchronized. • `<conn_user>` – User name of the user in the connection to be synchronized. • `<conn_group>` – Group name of the group in the connection to be synchronized.
Description	Triggers synchronization of the Managed System with SailPoint.
Example	The following statement triggers synchronization of the USER details between SailPoint and the Managed System. `CTSASYNC <token> USER USER1`

Maintenance

The Connector for ACF2 provides general maintenance procedures for the administrator's use. These procedures are run by submitting batch jobs.

The available procedures are:

Formatting the Diagnostic Level Dataset **103**
Displaying Local Connector for ACF2 Data **103**
Setting Transmitted Data Encryption **104**
Setting Stored Data Encryption **105**
Formatting the Offline Interceptor Dataset **107**
Recovering the Offline Interceptor After Failure **107**
Initializing the Connector Queue **109**
Changing the Size of the Connector Queue **109**
Printing the Connector Queue **110**
Renaming a Managed System **110**
Filtering Interception Messages **112**
Interception Acknowledgment **113**
Maintain Custom Field-Related Keywords **114**
Remove Custom Fields Support **114**

Formatting the Diagnostic Level Dataset

This procedure formats the diagnostic level dataset and should only be used at the request of Technical Support.

Member CTSDIAG in the Connector JCL library is a sample job which activates this procedure. The M= parameter refers to the relevant diagnostic flags member, which can be CTSACS, CTSACD or CTSAONI. These members are in the PARM library and should be updated with relevant flags based on requests from Technical Support.

Displaying Local Connector for ACF2 Data

This procedure is activated by member STATUS1 in the Connector JCL library. When STATUS1 job is submitted, the following Connector for ACF2 information is written to the job's sysout:

> **Caution**
> This procedure creates a report containing information about the local Connector for ACF2 instance. This job should only be used at the request of Technical Support.

- SMP's PTFs
- RSSPARM
- CTSPUSR
- RSSAPI
- RSSTABL
- RSSKWDS
- CTSPARM

Setting Transmitted Data Encryption

> **Note**
> Transmitted Data Encryption is not supported for communication with IdentityNow.

All data transmitted between Connector for ACF2 and IdentityIQ is (optionally) encrypted using an encryption key from the Encryption Key file.

Since the transmitted data must be understood by IdentityIQ and by Connector for ACF2, the Encryption Key file in both systems must match.

If encryption was activated during Connector for ACF2 installation, a synchronized Encryption Key file already exists. If you wish to change the Encryption Key file, it must be changed from IdentityIQ and then synchronized with Connector for ACF2.

Similarly, the enabling or disabling of Transmitted Data Encryption between IdentityIQ and Connector for ACF2 must be synchronized in both systems.

> **Note**
> Procedures for changing the encryption key and for enabling/disabling data encryption are described in the *IdentityIQ Administration Guide*. These are two-part procedures in which part of the procedure is performed in IdentityIQ and part in Connector for ACF2.

Follow the procedures described in *IdentityIQ Administration Guide*. At the appropriate point in the procedure, you are instructed to perform the Connector for ACF2 side of the procedure as described in this section.

Change the Transmitted Data Encryption Key

This is a two-part procedure in which part of the procedure is performed in IdentityIQ and part in Connector ACF2. Follow the procedures for encryption of security data in the *IdentityIQ Administration Guide*. At the appropriate point in that procedure, perform the supplemental steps listed below.

To change the Transmitted Data Encryption key for the Connector for ACF2 platform:

1. Stop the Connector for ACF2 Gateway and servers by specifying the following operator command:

   ```
   P CTSGATE
   ```

2. Transfer a copy of the Encryption Key file from IdentityIQ to Connector for ACF2 as described in *9.4 – Set Up Secured Communication* in 9 – Customize Communication Settings procedure.

3. Start the Connector for ACF2 Gateway (which automatically starts the Connector for ACF2 servers) by specifying the following operator command:

   ```
   S CTSGATE
   ```

Setting Stored Data Encryption

All data which is stored temporarily in Connector for ACF2 is encrypted using a Stored Data Encryption key (which differs from the Transmitted Data Encryption key). For example, sensitive security information that is written by the Interceptors to the Connector queue file is encrypted using the Stored Data Encryption key.

As part of the Connector for ACF2 installation procedure, an encryption key for stored data is created. However, the Stored Data Encryption key can be changed periodically to strengthen security.

> **Note**
> Before changing the encryption key, or before enabling or disabling Stored Data Encryption, verify that the Connector queue file does not contain data. This is because the Notification Server cannot process data in the Connector queue file which was encrypted by a previous key.

The Stored Data Encryption key is used internally by Connector for ACF2; there is no need to synchronize this key with SailPoint or any other Connector for ACF2 installation.

The following procedures are described below:

- Generating a new Stored Data Encryption key.
- Disabling (or enabling) the encryption of stored data.

Generate a New Stored Data Encryption Key

To generate a new Stored Data Encryption key:

1. Verify that the Connector queue dataset does not contain data using the procedure Printing the Connector Queue
2. Stop the Connector Online Interceptor by specifying the following operator command:

   ```
   P CTSAONI
   ```

3. Stop the Connector for ACF2 Gateway and servers by specifying the following operator command:

   ```
   P CTSGATE
   ```

4. Edit member CTSKGEN in the JCL library.
5. Submit the job and a new key is generated.

   All job steps must end with a condition code of `0`.

6. Start the Connector for ACF2 Gateway (which automatically starts the Connector for ACF2 servers) by specifying the following operator command:

   ```
   S CTSGATE
   ```

7. Start the Connector Online Interceptor by specifying the following operator command:

   ```
   S CTSAONI
   ```

Disable or Enable Stored Data Encryption

To disable (or enable) the encryption of stored data:

1. Verify that the Connector queue dataset does not contain data. (using procedure Printing the Connector Queue).
2. Stop the Connector Online Interceptor by specifying the following operator command:

   ```
   P CTSAONI
   ```

3. Stop the Connector for ACF2 Gateway and servers by specifying the following operator command:

   ```
   P CTSGATE
   ```

4. Edit member CTSPRSV in the Connector PARM library.
5. Set the ENCR_INT_ACT parameter to one of the following settings:
   - **N** to disable Stored Data Encryption.
   - **Y** to enable Stored Data Encryption.
6. Save the member.

7. Start the Connector for ACF2 Gateway (which automatically starts the Connector for ACF2 servers) by specifying the following operator command:

```
S CTSGATE
```

8. Start the Connector Online Interceptor by specifying the following operator command:

```
S CTSAONI
```

Formatting the Offline Interceptor Dataset

This procedure formats the Offline Interceptor dataset. This procedure is used during installation and to initialize the Offline Interceptor dataset if it was deleted and re-allocated.

Member FORMOFLI in the Connector JCL library is a sample job which activates this procedure. See Starting the Offline Interceptor Manually.

Recovering the Offline Interceptor After Failure

The Offline Interceptor operates in one of the following modes:

- **Init Mode** – The Offline Interceptor typically runs in Init mode the first time it is activated. In this mode, Offline Interceptor IMG files are created. These files contain an image of the security objects that are currently defined in ACF2. When operating in Init mode, the Offline Interceptor does not send any events to IdentityIQ.
- **Standard Mode** – The Offline Interceptor typically runs in Standard mode each time it is activated after its initial run. In this mode, the Offline Interceptor compares the current snapshot of ACF2 security objects to that in the IMG files. The differences represent security events and are sent to IdentityIQ.

By default, the Offline Interceptor operates in Init mode if the IMG files do not exist, and operates in Standard mode if the IMG files exist.

In the event an Offline Interceptor failure occurs, (for example due to ACF2 shutdown or manual cancellation), the IMG files may be incomplete or corrupt. To rectify this situation, the Offline Interceptor must be forced to run again in Init mode.

Force the Offline Interceptor to Run in Init Mode

> **Caution**
> SailPoint recommends that you consult with Technical Support before using this procedure.

Use one of the following methods to rebuild the IMG files:

Method A

1. Delete Offline Interceptor working and IMG files.
2. Delete all datasets with the following prefixes:
    - `%OLPREFS%`
    - `%OLVERS%`
    - `%RSSNAME%`
    - `OFL*`
3. Run the Offline Interceptor.

    This may be done manually or automatically. The Offline Interceptor will run in Init mode and creates new IMG files. (See Starting the Offline Interceptor Manually.)

Method B

Do one of the following:

- Change the CTSOFLI procedure parameter MODE value to -I and restart the Offline Interceptor.
- Specify the following command

```
S CTSOFLI,PARM='-I rssName'
```

> **Note**
> If you change the CTSOFLI procedure parameter MODE value to -I, be sure to change it back to its original value after restarting the Offline Interceptor.

- In IdentityIQ, perform Aggregation.

> **Note**
> The Aggregation operation should be done as soon as possible after the completion of the Offline Interceptor to ensure that the IMG files and IdentityIQ are synchronized. Until the Aggregation is performed, ACF2 security commands (such as GRANT/REVOKE or CREATE/DROP) that were issued between the failed Offline Interceptor run and the latest Init run of the Offline Interceptor are not updated in IdentityIQ.

Any future run of the Offline Interceptor will automatically be performed in Standard mode.

Initializing the Connector Queue

This procedure is used to initialize the Connector queue during the installation process. It is also used to initialize the Connector queue dataset if the dataset was deleted and re-allocated.

1. Stop all the Connector for ACF2 and Online Interceptor processes. Ensure that all the Connector for ACF2 and Interceptor processes have shut down.
2. Submit FORMQUE member in the Connector JCL library (a sample job which activates this procedure).
3. Restart the Connector for ACF2 and the Online Interceptor.

Changing the Size of the Connector Queue

Depending on the level of activity in the Managed System managed by Connector for ACF2, it may become necessary to change the size of the Connector queue. This section describes how to change the queue size.

> **Note**
> You can change the size of the Connector queue dataset regardless of whether or not it contains data. Data contained in the queue is preserved during this procedure.

To change the size of the Connector queue:

1. Stop the Connector Online Interceptor by specifying the following operator command:

   ```
   P CTSAONI
   ```

2. Stop the Connector for ACF2 Gateway and servers by specifying the following operator command:

   ```
   P CTSGATE
   ```

3. Ensure that all the Connector for ACF2 and Interceptor processes have shut down.
4. Allocate a new Queue dataset with increased DASD space, using a dataset name suffix other than ".QUEUE".

   For example if your existing Queue dataset is "CTSA.V400.QUEUE", allocate the new Queue dataset as "CTSA.V400.QUEUEN2".

   > **Note**
   > For sample allocation of a Queue dataset, see job FORMCTS in the Connector INSTALL library.

5. Edit sample job CTSQCR in the Connector JCL library, and specify the dataset name suffix for the new Queue (example "QUEUEN2").

This job consists of two steps:

a. Step 1 calls the CTSAQCR JCL procedure which calls the CTSQCR utility with appropriate parameters. The utility expects the new Queue dataset to be pre-defined and allocated. The utility formats the new Queue dataset and then copies the contents of the active Queue dataset to the new Queue dataset.

b. Step 2 calls the IDCAMS utility to change the suffix of the active Queue dataset (for example, to ".OLDQUEUE") so that it is no longer active, and to change the suffix of the new Queue dataset so that it becomes the active Queue dataset. Renaming of datasets can also be done using other methods available under OS/390, such as TSO or ISPF commands.

6. Submit the job.

7. Upon the successful termination of job CTSQCR and dataset rename commands, restart the Connector for ACF2 and the Online Interceptor. The new Queue dataset is now active.

Printing the Connector Queue

This procedure prints the contents of the Connector queue dataset.

Member PRTQUE in the Connector JCL library is a sample job which activates this procedure.

Renaming a Managed System

You can change the name assigned to the Managed System during Connector for ACF2 installation.

Once the Managed System has been defined in SailPoint, renaming the Managed System involves changes both on the SailPoint workstation and in Connector for ACF2. The procedure described below changes the name of the Managed System only in Connector for ACF2.

> **Caution**
> This procedure should only be performed within the framework of the procedure for changing the name of the Managed System in SailPoint.

The name of an Managed System is changed in Connector for ACF2 using utility CTSAADPT.

Member CTSADAPT in the JCL library contains a sample job to invoke the JCL procedure for the CTSAADPT utility (procedure <prefix>AADPT, where <prefix> is the prefix of your Connector JCL procedures). The utility should be run when Connector for ACF2 and the Interceptors are inactive.

Specify the old and new Managed System names as values of parameters FROMRSS and TORSS respectively when invoking the <prefix>AADPT procedure to execute the CTSAADPT utility.

This will change all occurrences of the old Managed System name in Connector for ACF2 datasets to new Managed System name. Note that the RSSPARM member itself is also modified.

The utility reports which datasets (and how many records in each) were modified. Most datasets considered for modification are allocated via JCL (see the relevant DD statements in the <prefix>AADPT JCL procedure).

One exception to this is the OFLRIMG dataset, used by Offline Interceptor utility. The DSNAME for this dataset is determined during CTSAADPT execution by the value of the RSS_WORK_DIR parameter for TORSS name in the RSSPARM member. The DSNAME dynamically allocated is the concatenation of the value of RSS_WORK_DIR with the suffix OFLRIMG (for example: CTSA.V400.MYRSS.OFLRIMG). If this DSNAME does not exist, no dynamic allocation (and thus, no modification) is done by CTSAADPT for this dataset.

Before running the CTSAADPT utility, verify that the RSS_WORK_DIR parameter value conforms to MVS dataset naming conventions.

> **Note**
> If the new Managed System name is longer than 8 characters, additional adjustments must be performed. For more information, see 11 – Adjust for Longer Managed System Names.

The CTSAADPT utility does not modify the Connector for ACF2 started task procedures. After running the utility, the following additional changes must be done manually.

Changes to Online Interceptor

Change the name of the Managed System in the CTSAONI (Online Interceptor) started task procedure as follows:

1. Stop all the Connector for ACF2 and Online Interceptor processes. Ensure that all the Connector for ACF2 and Interceptor processes have shut down.
2. Edit the CTSAONI member in the PROCLIB library containing Connector for ACF2 procedures.

   The name of the PROCLIB library is defined in variable **%PROCLIB%** of member DEFPARMS in the INSTALL library.
3. Locate the following line:

   ```
   Managed System=<rss_name>
   ```
4. Modify the value for `<rss_name>` to the new Managed System name.
5. Save the member.
6. Restart the Connector for ACF2 and the Online Interceptor.

> **Note**
> If you use the Delayed Delete utility , repeat step 2 through step 3 for procedure CTSC100. (see Renaming a Managed System and Shared ACF2 Database Support for more information.

Changes to Offline Interceptor

Change the name of the Managed System in the CTSAOFI (Offline Interceptor) procedure as follows:

1. Edit member CTSAOFI in the PROCLIB library containing Connector for ACF2 procedures.

   The name of the PROCLIB library is defined in variable **%PROCLIB%** of member DEFPARMS in the INSTALL library.

2. Locate the following line:

   ```
   Managed System=<rssName>
   ```

3. Modify the value for <rssName> to the new Managed System name.

4. Save the member.

Filtering Interception Messages

Events affecting the Managed System database that are intercepted by Connector for ACF2 are recorded in Connector for ACF2 log files. However, not all intercepted events are relevant for Connector for ACF2. You can set RSSPARM parameters in the member RSSPARM to filter the interception messages that are recorded in the log files.

The following Managed System parameters from the member RSSPARM are used to control filtering of interception messages:

LOG_INTERCEPT_MSG

Specifies the types of interception messages to be recorded in the CD log file.

Possible values for this parameter are described in the following table.

Value	Description
ALL	All messages are recorded, indicating in each case whether the intercepted event was sent to SailPoint. Default.
ACCEPTED	Only Managed System data included in aggregation (and therefore sent to SailPoint) is recorded.
IGNORED	Only Managed System data not included in aggregation (and therefore not sent to SailPoint) is recorded.
NONE	No messages are recorded.

LOG_GET_MSG

Specifies the filtering option for intercepted messages generated by the synchronization action (Managed System Retrieval Transaction). The messages are recorded in the CS log file.

Possible values for this parameter are described in the following table.

Value	Description
ALL	All Sync messages are recorded. Default.
NONE	No Sync messages are recorded.

To filter interception messages:

1. Stop the Connector for ACF2.

   See Shutting Down the Connector for ACF2 for more information.

2. Edit the RSSPARM member in the Connector PARM library.

3. Insert or modify either or both of the following parameters as necessary.

```
rss_name    LOG_INTERCEPT_MSG        ALL      <== Modify as required
rss_name    LOG_GET_MSG              ALL      <== Modify as required
```

4. Save the member and exit.

5. Restart the Connector for ACF2.

Interception Acknowledgment

The Connector for ACF2 sends intercepted Managed System events to IdentityIQ. When the *Interception Acknowledgment* function is active, IdentityIQ sends acknowledgment for events received to the Connector for ACF2. This process is managed by the **INTERCEPT_SEND_MAX** parameter in the RSSPARM file. This parameter determines whether or not Connector for ACF2 waits for acknowledgment from IdentityIQ for each intercepted Managed System event sent before sending the next event.

- When the **INTERCEPT_SEND_MAX** parameter is not present in the RSSPARM file, or is set to `0`, the interception acknowledgment mechanism is disabled. Connector for ACF2 sends events without waiting for acknowledgment.
- When **INTERCEPT_SEND_MAX** is set to 1, Connector for ACF2 waits for acknowledgment after each event is sent.

To set Interception Acknowledgment:

1. Stop the ACF2 connector.

   See Shutting Down the Connector for ACF2 for more information.

2. Edit member RSSPARM in the Connector PARM library.

```
ALL_RSS    INTERCEPT_SEND_MAX     1
```

3. Save the member and exit.

4. Restart the ACF2 connector.

Maintain Custom Field-Related Keywords

When changes are done to Custom fields definitions in ACF2 (new Custom Fields are added or Custom Fields are altered or deleted), the Custom Fields related keywords in the Account or Group entities should be re-defined according to the new Custom Fields definitions.

To re-define the Custom Fields related keywords:

1. Stop the Connector Gateway and Connector servers by specifying the following command:

```
P CTSGATE
```

2. Restart the Connector Gateway and Connector servers by specifying the following command:

```
S CTSGATE
```

3. Update Application schema in SailPoint with the new custom fields.

4. Aggregate all Accounts and Groups in order to retrieve the new Custom Fields data.

Remove Custom Fields Support

If there is a need to stop handling Custom Fields data via Connector and SailPoint, the Custom Fields support can be removed. The removal process includes removing the Custom Fields related keywords defined for Account and/or Group and removing all the Custom Fields data from accounts and/or groups.

To remove support for Custom Fields:

1. Stop the Connector gateway and Connector servers by specifying the following command:

```
P CTSGATE
```

2. Edit the RSSPARM member in the Connector PARM library.

3. Change the value in the **CUSTOM_FIELDS_SUPPORT** line to be `N`:

```
managedSystemName CUSTOM_FIELDS_SUPPORT N
```

4. Save the member.
5. Restart the Connector gateway and Connector servers by specifying the following command:

```
S CTSGATE
```

6. Delete all custom fields from application schema in SailPoint.
7. Aggregate all Accounts and Groups in order to remove the Custom Fields data.

Appendix A: Maintaining Connector for ACF2 using SMP/E

The design of the Connector for ACF2 SMP/E implementation accomplishes the following:

- Having as few modifications to the existing installation process as possible.

  This is achieved by enhancing the installation process to load the SMP/E dataset from tape into datasets with customizable names, and establish a complete SMP/E environment.

- Providing the user with an SMP/E environment that represents, as closely as possible, the target system, and that requires very few local modifications and tailoring during the Connector for ACF2 installation.

  This is achieved by shipping a complete set of SMP/E datasets, representing the environment being installed by the Connector for ACF2 installation process. These datasets are pre-loaded with information representing the environment installed. There is no need to perform any of SMP/E's traditional installation steps (RECEIVE/APPLY/ACCEPT) to perform the Connector for ACF2 installation. Upon completion of the installation process, the SMP/E datasets loaded represent the installed Connector for ACF2 product environment.

Packaging of Connector for ACF2 using SMP/E **116**

Connector for ACF2 Maintenance Procedures **118**

Packaging of Connector for ACF2 using SMP/E

The Connector for ACF2 product is shipped using a pre-installed SMP/E environment. The environment shipped includes SMP/E's CSI, service datasets, and distribution and target libraries. All these datasets are unloaded during the Connector for ACF2 installation jobs.

SMP/E parameters that are environment dependent are modified by the Connector for ACF2 installation process to contain the values specified by the user. When the installation process is complete, the SMP/E CSI and related datasets are customized to reflect the site environment.

Following the customization step, you will find a complete SMP/E environment, containing the Global zone, a target zone and a distribution zone.

The DDDEF names for the target and distribution libraries may be seen in Appendix C: Connector for ACF2 Datasets and JCL Procedures.

Zone Structure

The Connector for ACF2 product environment supplied contains three zones in a single CSI. These zones are:

- GLOBAL – The SMP/E global zone
- CTSATZN – Connector for ACF2 product target zone
- CTSADZN – Connector for ACF2 product distribution zone

All three zones are contained in a single VSAM KSDS cluster, in a single CSI structure. After the CSI has been loaded and customized by the Connector for ACF2 installation process, you will be able to move the loaded zones to other CSIs or rename these CSIs to suit their local standards.

Functions Installed

In the SMP/E environment supplied, the functions installed are:

Function	Description
CACF400 CRCF400 CTSA400 CTSS400	Connector for ACF2 elements
ECA7000 IOA700C IOA700E	Connector for ACF2 Gateway elements (Level 7.0.0)

Maintenance

Fixes supplied for Connector for ACF2 are module or other element replacement fixes and are shipped in PTF format.

Possible changes in ACF2 commands usage introduced by maintenance include:

- The Connector for ACF2 performs changes in a CA- ACF2 by issuing ACF2 commands (for example: INSERT, CHANGE, DELETE).
- Often a patch (fix or enhancement) to Connector for ACF2 adds usage of new ACF2 commands, or usage of new operands to the ACF2 commands currently used.
- Since CA- ACF2 allows a ACF2 administrator to control who is permitted to issue various ACF2 commands and their operands, it is important for z/OS sites using the Connector for CA- ACF2 to be aware and prepare/adapt the capabilities of the ACF2 users under which Connector for CA- ACF2 issues the above CA- ACF2 commands.

- The Connector for ACF2 issues commands to ACF2 using Managed System administrator. This is the ACF2 user defined to handle provisioning operations performed from SailPoint. This administrator should have SECURITY privilege.
- If you have any user-defined fields with privileges not assigned to this administrator, then you must select all of these privileges and assign them to the administrator.

  For example, if a user-defined field has **ALTER = ACCOUNT+SECURITY**, then administrator must be assigned with ACCOUNT and SECURITY privileges.

Connector for ACF2 Maintenance Procedures

This section describes the Connector for ACF2 maintenance procedures in detail.

Running SMP/E Jobs

The Connector for ACF2 installation supplies a JCL procedure (CTSASMP) designed to run SMP/E, and allocate its CSI, auxiliary datasets, and target and distribution libraries. This procedure should be used for all SMP/E operations involving Connector for ACF2.

> **Note**
> The samples below are only intended to demonstrate the most basic use of each SMP/E command; they are not intended to provide the complete format of each command.

For a full description of SMP/E commands, refer to the appropriate SMP/E documentation, especially the SMP/E User's Guide and SMP/E Reference. Full publication names and IBM Form Numbers are provided in the References section at the end of this appendix.

SMP/E requires the user to define, for each operation, the zone for which the operation will take effect. This definition is done via the SET BOUNDARY SMP/E command. This command must be the first in SMP/E's command stream, and is effective for all subsequent commands up to the next SET command.

Receiving Maintenance

The SMP/E RECEIVE command loads SYSMODs into SMP/E's Global zone. The RECEIVE command does not update any element, and its major purpose is to store the SYSMOD in the Global zone for subsequent processing. The following is a sample RECEIVE job:

```
//jobname JOB jobparms//RECEIVE EXEC CTSASMP
  SET BOUNDARY (GLOBAL).
  RECEIVE [SYSMODS] [SELECT(sysmod1,sysmod2,...)].
/*
//SMPPTFIN DD ...
```

The SMPPTFIN DD statement should point to a sequential dataset (or PDS member) holding the SYSMODs to be processed.

The SELECT parameter of the RECEIVE command can be used to limit its scope to the SYSMODs specified in it. Omitting this parameter will make SMP/E receive all the SYSMODs contained in the dataset pointed to by the SMPPTFIN DD statement. Omitting the SYSMODS parameter will cause SMP/E to attempt and receive HOLD information contained in the SMPHOLD dataset.

> **Note**
> The SMPRECIV member in the Connector JCL library contains JCL tailored to RECEIVE PTFs for Connector for ACF2.

Applying Maintenance

After the maintenance is received, it must be installed into the software product to take effect. The installation is done via the APPLY command. The following is a sample APPLY job:

```
//jobname JOB jobparms//APPLY  EXEC CTSASMP
  SET BOUNDARY(CTSATZN).
  APPLY [SELECT(sysmod1,sysmod2,....)].
/*
//
```

> **Note**
> The SMPAPPLY member in the connector's JCL library contains JCL tailored to APPLY PTFs for Connector for ACF2.

Accepting Maintenance

Following the SYSMODs application into the target zone, and after sufficient time has passed, the SYSMOD should be ACCEPTed into the distribution zone. The following is a sample ACCEPT job:

```
//jobname JOB jobparms//ACCEPT EXEC CTSASMP
  SET BOUNDARY(CTSADZN).
  ACCEPT [SELECT(sysmod1,sysmod2,....)].
/*
//
```

> **Note**
> The SMPACCPT member in Connector JCL library contains JCL tailored to ACCEPT PTFs for Connector for ACF2.

Producing SMP/E Reports

It is recommended that you ACCEPT the maintenance after a certain period of time has elapsed since the maintenance was applied. To verify which SYSMODs have been applied and are not ACCEPTed yet, run the following sample job:

```
//jobname JOB jobparms//REPORT EXEC CTSASMP
  SET BOUNDARY(GLOBAL).
  REPORT SYSMODS INZONE(CTSATZN) COMPAREDTO(CTSADZN).
/*
//
```

Appendix B: Connector for ACF2 Configuration Parameters

This appendix describes configuration parameters used by Connector for ACF2. Many of these parameters can be modified to suit user requirements.

The Connector for ACF2 configuration parameters are stored in the following members in the Connector PARM library:

Member	Description
CTSPUSR	Parameters in this member are common to all Connector for ACF2 platforms. For example, this member includes the parameter that determines whether or not Transmitted Data Encryption is enabled.
RSSPARM	Parameters in this member are specific to ACF2. For example, this member includes the parameter that determines the interval between runs of the Offline Interceptor.
RSSAPI	This member contains all Connector for ACF2 calls and the corresponding script-related parameters.
CTSPARM	Parameters in assembler format which require compile in case they are updated.

The parameters in these members are set during the Connector for ACF2 installation and customization process and should not be modified.

The contents of these members are described in this appendix.

CTSPUSR – Connector for ACF2 Parameters **121**

RSSPARM – Managed System Parameters **122**

RSSAPI – Connector Entries and Scripts **133**

CTSPARM – Assembler Format Parameters **133**

CTSPUSR – Connector for ACF2 Parameters

The following table describes the CTSPUSR parameter.

Parameter	Description
WRITE_TO_QUEUE	Whether the messages of account and group aggregations are written to the Queue file. Default – N

Parameter	Description
	Note MAIN_CS MUST be specified in column 1 of CTSPUSR with WRITE_TO_QUEUE parameter.

RSSPARM – Managed System Parameters

This section contains a description of parameters in the RSSPARM member, followed by a listing of the member as it appears after installing the Connector for ACF2.

Description of Parameters

Each parameter in the RSSPARM member is applicable either for all MSCSs managed by the Connector for ACF2 installation or for a specific Managed System.

Each parameter in the RSSPARM member has the following syntax:

```
rss parameterName value
```

where:

- **rss** – Name of the Managed System to which the parameter applies. If the parameter applies to all Managed System, contains ALL_RSS.
- **parameterName** – Name of the RSSPARM parameter.
- **value** – Value assigned to the parameter.

The tables that follow describe the parameters that can appear in the RSSPARM member.

> **Note**
> Many of the parameters in the tables are not automatically present in the RSSPARM file after Connector for ACF2 installation. If you wish to assign a value to a specific parameter, it may be necessary to add the parameter to the file. The value labeled as *Default* appearing in the Values column of the tables that follow indicates the value assigned if the parameter is *not* present in the member or if the parameter is assigned an invalid value. To see the default value for parameters that *are* present in the member, see the listing of the RSSPARM members in General Parameters.

Each table contains the following columns:

Column name	Description
Parameter	Name of the RSSPARM parameter. The presence of the symbol * in this column indicates that if the parameter is assigned an invalid value, Connector for ACF2 automatically assigns the parameter the "default" value specified (see the description of the **Values** row below). The presence of "(ALL_RSS)" in this column indicates that the parameter is applicable to all MSCSs managed by the Connector for ACF2 installation. If the parameter is specific to a certain type of Managed System, the parameter is applicable to all MSCSs of that type.
Description	Description of the parameter.
Values	Possible parameter values, or limitations. Where specified, the **Default** value in this column indicates the value assigned if the parameter is not present in the RSSPARM member or if the parameter is assigned an invalid value. To see the default value (if any) for a parameter, locate the parameter in the General Parameters.

General Parameters

The following table contains descriptions of RSSPARM parameters which are specified once for all MSCSs managed by the Connector for ACF2 installation. Each parameter name is preceded by ALL_RSS.

Parameter	Description	Value
CHECK_SYNC_OBJS	During aggregation – Number of entity/connection operations handled by Connector for ACF2, after which an "active" confirmation message is sent to SailPoint.	Default – 100
INTERCEPT_SEND_MAX	When INTERCEPT_SEND_MAX is set to a positive numeric value, Connector for ACF2 waits for acknowledgment after the amount of events specified are sent. For example, when INTERCEPT_SEND_MAX is set to 10, Connector for ACF2 sends 10 event messages to IdentityIQ before waiting for an acknowledgment message.	**0** – Do not wait (Default) **1 or more** – Number of events which are sent to IdentityIQ before waiting for an acknowledgment from IdentityIQ.
OFLI_VERBOSE	Whether the Log Message of the Online and Offline Interceptor is sent to Managed System console.	Y, N Default – N
OCCUPIED_QUEREU_DATA	For future use. Do not change the default value.	Y, N

Parameter	Description	Value
		Default – N
STAT_CHKSUM_INTRVL	During aggregation – Number of entity/connection checksums received by Connector for ACF2 from SailPoint, after which a message is sent to the Connector for ACF2 log and an event is sent to SailPoint.	Default – 5000
STATIST_INTRVL	During aggregation – Number of entities or connections received by Connector for ACF2 from SailPoint, after which a message is sent to the Connector for ACF2 log and an event is sent to SailPoint.	Default – 20
STATUS_INTERVAL	When a Managed System is not active, interval at which Connector for ACF2 checks the status of the Managed System. When the Managed System is active, an event is sent to SailPoint.	Format – hhmmss Default – 000500 (5 minutes)
STOP_REQ_MSGS	During aggregation – Number of entity/connection operations handled by Connector for ACF2, after which an "active" confirmation message is sent to SailPoint.	Default – 10
WAIT_LOCK	Wait Lock (seconds)	Default – 60
WAIT_QUEUE	Wait Queue (seconds)	Default – 60

Managed System-Specific Parameters

The following table contains descriptions of parameters which are specified separately for each individual Managed System managed by the Connector for ACF2 installation. The name of the Managed System must appear before the parameter name in the record.

Parameters	Description	Values
ACF2_GROUPS	Whether connection events are intercepted by Online and Offline Interceptors. (May be relevant for customers who do not define Groups in IdentityIQ for the ACF2 Connector).	Y, N Default – Y
ADMIN_CASE_SENS*	Whether the Administrator name is case-sensitive.	Y, N Default – Y
ADMIN_USER_REQ*	Whether a default administrator is used.	Y, N Default – N
CUSTOM_FIELDS_SUPPORT	Whether product supports ACF2 custom fields feature.	Y, N

Parameters	Description	Values
		Default – N
DEFAULT_ADMIN	Name of Connector for ACF2 default administrator account, which is used for GET operations. Only applicable when ADMIN_USER_REQ=Y.	
DEFAULT_CD_ADMIN	Name of default administrator for the CD process. If special administrator for CD process is not required, then DEFAULT _ADMIN is used. Only applicable when ADMIN_USER_REQ=Y.	
DEFAULT_CS_ADMIN	Name of default administrator for the CS process. If special administrator for CS process is not required, then DEFAULT _ADMIN is used. Only applicable when ADMIN_USER_REQ=Y.	
DEFAULT_OFLI_ADMIN	Default administrator for the Offline Interceptor process. If special administrator for Offline Interceptor process is not required, then DEFAULT _ADMIN is used. Only applicable when ADMIN_USER_REQ=Y.	
DELETE_INTERCEPT_CHECK	When a delete event is detected, whether the Notification server should call a **get** function to check that an object was actually deleted on the Connector for ACF2 platform. If the call determines that the object exists, an update event is sent to IdentityIQ instead of a delete event.	Y, N Default – N
DISREGARD_RU_SUSPENDED	Enables the prevention of restore requests coming from Password Manager from being executed by Connector for ACF2.	Y,N Default – N
LOG_GET_MSG	Filtering option for messages generated by the synchronization action-Managed System Retrieval Transaction). The messages are recorded in the Transaction Server (CS) log file.	ALL – All Sync messages are written to the CS log file. NONE – No Sync messages are written to the CS log file. Default – ALL

Parameters	Description	Values
LOGIN_INTERCEPT	Determines whether TSO LOGON events are intercepted by Online and Offline Interceptors.	Y, N Default – Y
LOG_INTERCEPT_MSG	Types of interception messages to be recorded in the Notification Server (CD) log file.	ALL – All messages are recorded, indicating in each case whether the event was sent to IdentityIQ ACCEPTED – Only Managed System data included in the aggregation (and therefore sent to IdentityIQ) is recorded. IGNORED – Only Managed System data not included in the aggregation (and therefore not sent to IdentityIQ) is recorded. NONE – No messages are recorded. Default – ALL
MAX_Q_TRY	When the Queue file is full, the Offline Interceptor may retry to write to it according to the number set in MAX_Q_TRY in RSSPARM.	Values – Any number Coded – Forever (retries continue indefinitely.)
MAX_Q_TRY	When the Queue file is full, the Offline or the Online Interceptor may retry to write to the Queue file according to the value set in this parameter.	Value for number of retries. If this parameter is not set the default is to keep retrying without any limit.
OFLI_INTERCEPT	Whether the Offline Interceptor is started automatically by the Notification Server.	Y – The Offline Interceptor is started periodically by the

Parameters	Description	Values
		Notification server. N – The Offline Interceptor is not started by the Notification server. You must provide another means of scheduling the Offline Interceptor. Default – Y
OFLI_INTERVAL*	Minimum interval between consecutive activations of the Offline Interceptor.	Value in the format **hhmmss** Default – 010000
OFLI_MAX_DELETE_PERCENT	The maximum number (expressed as a percent) of existing objects that you realistically expect would be deleted between consecutive executions of the Offline Interceptor. If the Offline Interceptor determines that the number of deleted objects exceeds this threshold, the Offline Interceptor stops operation and places an error message in its message file. The purpose of this parameter is to detect situations where a large number of objects are temporarily unreachable and therefore appear to have been deleted. If the number of such "deleted" objects exceeds the threshold, the Offline Interceptor will not send delete events to SailPoint.	Default – 100 (no limit on objects deleted)
ONLI_EVENT_USER_PWD_ONLY	This parameter controls whether user and group events are intercepted and sent by Online Interceptor. By default it is set to N, meaning that all users, groups, connections and password events are sent to SailPoint. When set to Y, only the password events are sent to SailPoint. In this case the password violation events controlled by SEND_PASS_VIOLATION parameter are not sent	Y, N

Parameters	Description	Values
	to SailPoint.	
ONLI_MAX_EVENTS	Number of events which may be active in memory, when Queue file is full. Each entry holds 2,560 bytes in memory, depending on the event type (user, group, connection, password) and length of userid, group, password (above 16M line). A queue full situation might occur when Online Interceptor is active and CD component of the Connector for ACF2 is down or reads events from the Queue too slowly comparing to the written events by the Online Interceptor. When queue full situation occurs, Online Interceptor continues accepting events from the SMF exit and from the password exits and these events are accumulated in memory, until queue full situation is relieved. The number of events which can be accumulated in memory is determined by this parameter.	Default – 20000
ONLI_MIN_NOTIFY_EVENT%	Percent number of residual place for new events left in memory that below it, CTS4509W message will be sent by Online Interceptor, each time it handles a new event. The 100% is ONLI_MAX_EVENTS which its default is 20,000. When CTS4509W message is sent, Online Interceptor is able to handle only ONLI_MIN_NOTIFY_EVENT% more events from SMF exit and password exits. If this situation is not relieved fast enough, new events may get lost and not be handled by the Online Interceptor.	Default – 10% 2,000 by default, as ONLI_MAX_EVENTS is 20,000 by default.
ONLI_SEMAPHORE	The name of lock obtained during Online Interceptor operation.	Y, N Default – N

Parameters	Description	Values
	Valid only for Managed System that support the Online Interceptor.	
PASS_PASSWORD*	Whether the password is passed to pre/post-scripts in update password transactions.	Y, N Default – N
PASSWORD_EVENT_FILTER	This parameter makes it possible to filter all password_change events and user_update events.	CTSRSS PASSWORD_ EVENT_FILTER NETLOCK or CTSRSS PASSWORD_ EVENT_FILTER NETL* where: NETLOCK and NETL* are the filtering job name and prefix.
RSS_TYPE	Type of Managed System.	ACF2
RSS_WORK_DIR	The prefix used to dynamically allocate working datasets.	Must conform to MVS dataset naming conventions.
SCRIPT_DIR	The name of a dataset containing customer scripts.	
SCRIPT_SEP_ENTRY	Separator entry value for list fields passed to scripts.	Default – comma (,)
SCRIPT_SEP_FIELD	Separator field value for list fields passed to the scripts.	Default – semicolon (;)
SEND_PASS_VIOLATION	This parameter controls whether password violation events are intercepted and sent by Online Interceptor. By default, it is set to Y (send the interception). By setting it to N, all these events are filtered out and not sent to IdentityIQ. Events are sent only if the ONLI_ EVENT_USER_PWD_ONLY parameter is not set to Y.	Y, N
SEND_PWD_TO_SCRIPT	Whether the SailPoint Administrator password is sent to scripts.	Y, N Default – N

Parameters	Description	Values
SEND_RSSPRM_TO_SCRIPT	Whether to send RSSPARM parameters to scripts.	Y, N Default – Y
SUSPEND_IS_LOCKED	The default (when SUSPEND_IS_LOCKED is not specified or when it's set to N) is that ACF2's SUSPEND relates to PassPort's Revoked or SailPoint RU_SUSPENDED keyword. When SUSPEND_IS_LOCKED is set to Y, ACF2's SUSPEND relates to PassPort's Locked or SailPoint RU_LOCKED keyword. In other words, when SUSPEND_IS_LOCKED is set to Y and ACF2 user has SUSPEND attribute in its LID record, the ACF2 Connector will set RU_LOCKED to Y when Sync Account is done or when interception occurs for this account. When SUSPEND_IS_LOCKED is set to Y and the Unlock request is done from PassPort, the ACF2 Connector will issue: `CHA user PSWD-VIO(0) NOSUSPEND ACF2 command.`	Y,N Default – N
SYNC_SEMAPHORE	Name of lock obtained while the Offline Interceptor or aggregation is running (in order to avoid concurrent execution).	
ONLI_EVENT_CONTAINER	Whether intercepted container events are sent to SailPoint.	Y, N Default – Y
ONLI_EVENT_GROUP	Whether intercepted group events are sent to SailPoint	Y, N Default – Y
ONLI_EVENT_USER	Whether intercepted user events are sent to SailPoint.	Y, N Default – Y
ONLI_EVENT_USER_PWD_ONLY	(Only relevant when ONLI_EVENT_USER = Y) Type of intercepted user events sent to SailPoint.	Y – Only user password change events are sent. N – All user events are sent. Default – N

Parameters	Description	Values
USAAPI_LIB_NAME	The name of an optional DD statement in the CS and CD STC procedure. This parameter is used for customized Connector for ACF2 that are called from a non-APF load library.	
WAIT_WHEN_Q_EOF	Number of seconds Offline Interceptor waits before trying to write to the Queue file.	Values – Any number Coded – 120 seconds (2 minutes) Also, see MAX_Q_TRY parameter.
WAIT_WHEN_Q_EOF	When the Queue file is full, the Offline or the Online Interceptor may retry to write to the Queue file according to the value set in the MAX_Q_TRY parameter. The WAIT_WHEN_Q_EOF parameter sets the number of seconds to wait between retries.	Number of seconds to wait.

RSSPARM Parameters for ACF2 Connector

The following table contains descriptions of RSSPARM parameters which only appear in Connector for ACF2.

Parameter	Description	Values
CTSA_ID (ALL_RSS)	An ID that uniquely identifies the Connector for ACF2 installation.	Default – CTSA
HANDLE_ABENDS (ALL_RSS)	Determines the behavior of Connector for ACF2 function CTSAPITerm if Connector for ACF2 abends. Relevant for custom Connector for ACF2s developed using the SDK for OS/390. The ability to call the CTSAPITerm function if Connector for ACF2 abends is useful when the Connector for ACF2 includes databases and files that should be properly disconnected.	Y – The recovery routines from the function **CTSAPITerm** are invoked. N – The standard SAS-C abend handler is invoked. The SAS-C abend handler only provides DUMP and diagnostic information and does not perform application-specific recovery. Default – N
MAX_SCRIPT_NOTIFY (ALL_RSS)	Number of entries in the Script Notify Buffer.	Default – 250

Parameter	Description	Values
OFLI_STCNAME	Offline Interceptor started task name.	Default – %PROCPREFS%AOFI
ONLI_ACSJBN	Name of the Transaction Server reported to the Online Interceptor.	Default – %PROCPREFS%ACS
ONLI_DETAIL_MSGS	Whether to retrieve detailed messages from the Online Interceptor.	Y, N Default – N
ONLI_DYNAM_NPX	Whether to load the ACF2 New Password Exit (ACFNPXIT) dynamically.	Y – Online Interceptor loads ACFNPXIT dynamically. N – ACFNPXIT is not loaded by the Online Interceptor. Default – N
ONLI_PASSWORD_CASE	Case in which the Online Interceptor sends the password of the administrator (who has performed a password update) to IdentityIQ.	LOWER – Send in lowercase. ASIS – Send as received (with no translation). UPPER – Same as ASIS. Default – LOWER
ONLI_PASSWORD_FILTER	Whether to send password updates to IdentityIQ.	SUPPRESS – password updates are not sent to IdentityIQ. FORWARD – password updates are sent to IdentityIQ. Default – FORWARD
RULE_BACKUP	Whether to build a backup command when creating a resource rule in IdentityIQ.	Y, N Default – N
SEND_PASS_VIOLATION	Whether an interception is sent to IdentityIQ when a user specifies an incorrect password while logging in to the Managed System.	Y, N Default – Y
SIMULATE_ARI	ACF2 rule update mode.	Y – Simulate mode (rule is not updated) N – Update mode (rule is updated) Default – N

Parameter	Description	Values
USE_MAXRULE	Whether to build the MAXRULE parameter in a ruleset definition. If a resource has more than 250 connected ACEs, it is recommended that this parameter be set to Y.	Y – MAXRULE equivalent to the actual ACF2 ruleset size is created. N – ACF2 MAXRULE default value is used. Default – N

RSSAPI – Connector Entries and Scripts

Each line in this member (excluding the comment lines) represents a Connector call which is followed by various parameters. The parameters influence the behavior of the Connector.

For a description of the parameters in each line, see the **Standard Managed System User Fields** table in User Data Translation Tables.

Pre-scripts and Post-scripts must be stored in the PDS library which was set in parameter SCRIPT_DIR of member RSSPARM. The default value for this parameter is:

```
prefix.version.USER.CLIST
```

where:

- **prefix** – Value set for parameter OLPREFS in member INSTALLD in the Connector INSTALL library.
- **version** – Value set for parameter OLVERS in member INSTALLD in the Connector INSTALL library.

CTSPARM – Assembler Format Parameters

This CTSPARM member in PARM library includes few parameters in an assembler source format. This means that, once a parameter is updated, the member must be saved and then it must be compiled and linked. This is done with the CTSPARMJ member located in the INSTALL library. So once CTSPARM is updated, CTSPARMJ must be submitted.

Parameter	Description
ENQRNL	When global resource serialization (GRS), (for example, ENQ or DEQ) encounters a request for a resource with a scope of SYSTEMS, it scans the SYSTEMS exclusion resource name list (RNL) to determine the scope of the requested resource. However, if the request specifies RNL-L=NO, GRS would not scan the SYSTEMS exclusion RNL

Parameter	Description
	and bypass the RNL search. By default, the CTSPARM ENQRNL parameter is set to **Y**. If the GRS environment requires that resource requests bypass the RNL search, set ENQRNL to **N**.
QNAME	Unique name used to protect access to the Queue file by Interceptors and Notification Server.

Appendix C: Connector for ACF2 Datasets and JCL Procedures

This appendix describes the following information.

Connector for ACF2 Dataset List **135**

Connector for ACF2 JCL Procedures **138**

Connector for ACF2 Dataset List

This section contains lists of datasets used by the Connector for ACF2.

Connector for ACF2 Installation Datasets

The following datasets are allocated during the Connector for ACF2 installation procedure. In the names of these datasets, **<prefix>** is the value specified for parameter **ILPREFS**, and **<version>** is the value specified for parameter **ILVERS**. Both parameters are located in LOADCTS member in the Connector INSTALL library.

> **Note**
> These parameters are set in 4.1 – Tailor member LOADCTS during installation step 4 – Allocate and Load Connector for ACF2 Installation Libraries using values set in Allocate and Load Connector for ACF2 Datasets ($LOADINS and LOADCTS Dobs).

Dataset	Description
<prefix>.<version>.CLIST	REXX library
<prefix>.<version>.CTRANS	SAS/C Runtime Load library
<prefix>.<version>.CMSG	Message text library
<prefix>.<version>.INSTALL	Installation library
<prefix>.<version>.ISMSGENG	ISPF English messages of CTSGATE ISMSGENG
<prefix>.<version>.JCL	Sample jobs library
<prefix>.<version>.LOAD	Load library
<prefix>.<version>. LOADE	SSL load modules LOADE
<prefix>.<version>.MAC	Macros library
<prefix>.<version>.MSG	Message text library
<prefix>.<version>.MSGENG	English messages of CTSGATE MSGENG
<prefix>.<version>.PARM	Parameters library

Dataset	Description
<prefix>.<version>.PANELENG	English panels of CTSGATE PANELENG
<prefix>.<version>.PROCLIB	JCL procedures library
<prefix>.<version>.SAMPLE	Sample REXX and REXX library
<prefix>.<version>.SECSRC	Sample security exits for CTSGATE (not used)
<prefix>.<version>.UPGRADE	UPGRADE jobs

Operation Datasets

The following datasets are allocated during the Connector for ACF2 installation procedure. In the names of these datasets, **<prefix>** is the value specified for parameter **OLPREFS**, and **<version>** is the value specified for parameter **OLVERS**. Both parameters are located in member LOADCTS in the Connector INSTALL library.

> **Note**
> These parameters are set in 4.1 – Tailor member LOADCTS during installation step 4 – Allocate and Load Connector for ACF2 Installation Libraries using values set in Allocate and Load Connector for ACF2 Datasets ($LOADINS and LOADCTS Dobs).

Dataset	Description
<prefix>.<version>.CARECNN	Managed user to group connections
<prefix>.<version>.CAREGRP	Managed groups
<prefix>.<version>.CAREOE	Managed organization elements
<prefix>.<version>.CAREUSR	Managed users
<prefix>.<version>.DIAGLVL	Diagnostics setup
<prefix>.<version>.ENCREXT	Transmitted Data Encryption
<prefix>.<version>.ENCRINT	Stored Data Encryption
<prefix>.<version>.GTWPARM	Gateway parameters
<prefix>.<version>.QUEUE	Interception Queue dataset
<prefix>.<version>.RSSKWDS	Managed System specific keywords table
<prefix>.<version>.RSSOFLI	Offline interceptor table
<prefix>.<version>.USER.CLIST	Scripts dataset
<prefix>.<version>.ACF2GDB	Groups database
<prefix>.<version>.ACF2UDB	Users database

SMP/E Distribution Datasets

The following datasets are allocated during the Connector for ACF2 installation procedure. In the names of these datasets, **<prefix>** is the value specified for parameter **SPDPREF**, and **<version>** is the value specified for parameter

SPDVER. Both parameters are located in member LOADCTS in the Connector INSTALL library.

> **Note**
> These parameters are set in 4.1 – Tailor member LOADCTS during installation step 4 – Allocate and Load Connector for ACF2 Installation Libraries using values set in Allocate and Load Connector for ACF2 Datasets ($LOADINS and LOADCTS Dobs).

Dataset	Description
<prefix>.<version>.ACMSG	CMSG Dlib
<prefix>.<version>.ACLIST	REXX Dlib ACLIST
<prefix>.<version>.AINSTALL	INSTALL Dlib
<prefix>.<version>.AISMSGEN	ISMSGENG Dlib AISMSGEN
<prefix>.<version>.AIOALOAD	GATEWAY LOAD Dlib
<prefix>.<version>.AJCL	JCL Dlib
<prefix>.<version>.ALOADE	SSL Load modules Dlib ALOADE
<prefix>.<version>.AMAC	Macro Dlib
<prefix>.<version>.AMSG	MSG Dlib
<prefix>.<version>.AMSGENG	English messages of CTSGATE Dlib AMSGENG
<prefix>.<version>.APARM	PARM Dlib
<prefix>.<version>.APROCLIB	PROCLIB Dlib
<prefix>.<version>.APANELEN	PANELENG Dlib APANELEN
<prefix>.<version>.ASAMPLE	SAMPLE Dlib
<prefix>.<version>.ASECSRC	SECSRC Dlib ASECSRC
<prefix>.<version>.AUPGRADE	UPGRADE Dlib AUPGRADE

SMP/E Datasets

The following datasets are allocated during the Connector for ACF2 installation procedure. In the names of these datasets, **<prefix>** is the value specified for parameter **SPAPREF**, and **<version>** is the value specified for parameter **SPAVER**. Both parameters are located in member LOADCTS in the Connector INSTALL library.

> **Note**
> These parameters are set in 4.1 – Tailor member LOADCTS during installation step 4 – Allocate and Load Connector for ACF2 Installation Libraries using values set in Allocate and Load Connector for ACF2 Datasets ($LOADINS and LOADCTS Dobs).

Dataset	Description
<prefix>.<version>.SMPLOG	SMP/E Work dataset

Dataset	Description
<prefix>.<version>.SMPLOGA	SMP/E Work dataset
<prefix>.<version>.SMPLTS	SMP/E Work dataset
<prefix>.<version>.SMPMTS	SMP/E Work dataset
<prefix>.<version>.SMPPTS	SMP/E Work dataset
<prefix>.<version>.SMPSCDS	SMP/E Work dataset
<prefix>.<version>.SMPSTS	SMP/E Work dataset

SMP/E CSI

The following dataset is allocated during the Connector for ACF2 installation procedure. In the names of this dataset, **<prefix>** is the value specified for parameter **SPCPREF**, and **<version>** is the value specified for parameter **SPCVER**. Both parameters are located in member LOADCTS in the Connector INSTALL library.

> **Note**
> These parameters are set in 4.1 – Tailor member LOADCTS during installation step 4 – Allocate and Load Connector for ACF2 Installation Libraries using values set in Allocate and Load Connector for ACF2 Datasets ($LOADINS and LOADCTS Dobs).

Dataset	Description
<prefix>.<version>.CSI	SMP/E CSI

Connector for ACF2 JCL Procedures

The following JCL procedures are copied to your system PROCLIB library during the Connector for ACF2 installation procedure.

JCL procedure	Description
CTSAADPT	Connector for ACF2 Managed System rename utility
CTSACD	Connector Notification Server (CD)
CTSACS	Connector Transaction Server (CS)
CTSADFR	Connector Offline Interceptor file formatting utility
CTSALERT	Connector for ACF2 alert data utility
CTSAONI	ACF2 Online Interceptor
CTSAQCR	Connector for ACF2 copy and format Queue dataset utility
CTSAQFR	Connector for ACF2 Queue formatting utility
CTSAQPR	Connector for ACF2 Queue printing utility

JCL procedure	Description
CTSASMP	Connector for ACF2 SMP/E procedure
CTSCDIAG	Connector for ACF2 diagnostics file formatting utility
CTSDIAGI	Connector for ACF2 diagnostic initialization utility
CTSGATE	Connector for ACF2 Gateway
CTSKGEN	Connector for ACF2 Stored Data Encryption utility
CTSKEYGEN	Connector for ACF2 Stored data encryption utility
CTSAOFI	Offline Interceptor
CTSUPMSG	Connector for ACF2 message update table

Appendix D: Copying a Connector for ACF2 Installation

This appendix describes the procedure to copy an existing installation of Connector for ACF2 from one MVS system to another.

The following terms are used throughout this appendix:

- **source system** – MVS system where Connector for ACF2 was originally installed.
- **target system** – MVS system to which Connector for ACF2 installation is copied.

Installation Copy Procedure **140**

Installation Copy Procedure

Use the procedure that follows to copy an existing installation of Connector for ACF2 from one MVS system to another.

> **Note**
> If LOCALCOPY value is set for the %PROCLIB% parameter in the DEFPARMS member in the Connector INSTALL library, the same procedures will be used by the source and new Connector for ACF2 environments. If the Managed System name has to be changed, contact SailPoint Customer Support for instructions. See 3 – Adjust Connector for ACF2 Parameters for more information.

1 – Copy Connector for ACF2 JCL Procedures

During installation of Connector for ACF2, various JCL procedures were copied to the system procedures library. The JCL procedures are used by Connector for ACF2 started tasks and by jobs activating the Connector for ACF2 utilities. These JCL procedures must be copied to the target system.

The procedures may be copied manually or using the instructions specified in installation step 5 – Tailor Connector for ACF2 Members with Site Parameters .

1A – Connector for ACF2 started tasks

The following JCL procedures are Connector for ACF2 started tasks and must be copied to the target system JCL procedures library:

- CTSACD
- CTSAOFI

- CTSACS
- CTSAOFS
- CTSGATE
- CTSAONI

When STCJOBs are used:

- The corresponding STCJOB members must be copied to the target system STCJOBS library.
- If LOCALCOPY value is set for the %PROCLIB% parameter in DEFPARMS member in the Connector INSTALL library, make sure the target system has access to the Connector for ACF2 PROCLIB library.

1B – Connector for ACF2 utilities

The following JCL procedures are used by Connector for ACF2 utilities and maintenance jobs. They are not required for the daily operation of Connector for ACF2. However, they are required for customization and installation operations performed in subsequent steps described in this procedure.

- CTSADFR
- CTSDIAG
- CTSAQFR
- CTSDIAGI
- CTSAQCR
- CTSAQPR
- CTSKGEN
- CTSALERT
- CTSUPMSG
- CTSAADPT

2 – Copy Connector for ACF2 Datasets

The following parameters are used in this step:

- **<i_prefix>** – Value set for parameter ILPREFS in member LOADCTS in the Connector INSTALL library.

  Default: CTSA.

- **<i_version>** – Value set for parameter ILVERS in member LOADCTS in the Connector INSTALL library.

  Default: V400.

- **<o_prefix>** – Value set for parameter OLPREFS in member LOADCTS in the Connector INSTALL library.

  Default: CTSA.

- **<o_version>** – Value set for parameter OLVERS in member LOADCTS in the Connector INSTALL library.

  Default: V400.

For more information regarding parameters in member LOADCTS, see step 4.1 – Tailor Member LOADCTS in installation step 4 – Allocate and Load Connector for ACF2 Installation Libraries.

2A – Operations Datasets

The following datasets are used during the operation of Connector for ACF2 and must be copied from the source system to the target system:

```
<i_prefix>.<i_version>.CLIST
<i_prefix>.<i_version>.CMSG
<i_prefix>.<i_version>.LOAD
<i_prefix>.<i_version>.MSG
<i_prefix>.<i_version>.PARM
<i-prefix>.<i-version>.CTRANS
```

2B – Installation Datasets

The following datasets are not used by Connector for ACF2 started tasks and utilities but are required for customization and installation operations performed later in this procedure:

```
<i_prefix>.<i_version>.INSTALL
<i_prefix>.<i_version>.MAC
<i_prefix>.<i_version>.JCL
<i_prefix>.<i_version>.SAMPLE
<o_prefix>.<o_version>.USER.CLIST
<i_prefix>.<i_version>.PROCLIB
```

3 – Adjust Connector for ACF2 Parameters

Perform the customization steps described below on the target system.

> **Note**
> As an alternative to steps 3A and 3B below, you can use utility CTSAADPT to rename the Managed System on the target system.
> For more information, see Renaming a Managed System.

3A – RSSPARM Parameters

Edit member RSSPARM member in the Connector PARM library.

The RSSPARM member contains the Managed System parameters for the installed Managed System.

Each line is in the format:

```
<Managed System-name> <parameter_name> <parameter_value>
```

where `<Managed System-name>` is either ALL_RSS or the Managed System name specified during installation.

If the Managed System name defined in SailPoint for the target system is different from the one specified in the member RSSPARM, you must update member RSSPARM to reflect that difference.

To perform the required update in member RSSPARM, change the <Managed System-name> value on all lines that specify the source system Managed System name to the new Managed System name that is correct for the target system.

In addition, the Managed System name qualifier in the value specified for parameter RSS_WORK_DIR must be modified.

> **Note**
> The Managed System name specified in the RSSPARM parameters file must match the name defined in SailPoint for the target system. If the names do not match, SailPoint will not be able to connect to Connector for ACF2.

3B – RSSAPI Parameters

Edit member RSSAPI member in Connector PARM library.

RSSAPI member contains the script activation definitions for the Managed System.

Each line is in the format:

```
RSS_type RSS_name additional_parameters
```

where:

- **RSS_type** – hyphen (-)
- **RSS_name** – Managed System name specified during installation.

  Default: MVSACF2

The complete syntax of this member is described under Structure of the RSSAPI Member.

If the Managed System name defined in SailPoint for the target system is different from the one specified in member RSSAPI, you must update member RSSAPI to reflect that difference.

To perform the required update in member RSSAPI, change the **RSS_name** parameter on all the lines containing the source system Managed System name to the new Managed System name that is correct for the target system.

3C – Adjust Procedures

If the Managed System name was changed, modify the `Managed System=` parameter in each of the following procedures:

- CTSAADPT
- CTSADFR
- CTSAOFI
- CTSAONI

3D – Allocate Connector for ACF2 Work Datasets

The datasets listed below are created during Connector for ACF2 installation. These datasets should *not* be copied from the source system to the target system. Instead, they should be allocated and formatted directly on the target system.

Originally, allocation of these datasets was performed by job FORMCTS, which is run in installation step 8 – Format Connector for ACF2 Datasets.

You may run job FORMCTS in the Connector INSTALL Library to allocate the datasets on the target system.

```
<o_prefix>.<o_version>.CARECNN
<o_prefix>.<o_version>.CAREGRP
<o_prefix>.<o_version>.CAREUSR
<o_prefix>.<o_version>.DIAGLVL
<o_prefix>.<o_version>.QUEUE
<o_prefix>.<o_version>.RSSKWDS
<o_prefix>.<o_version>.RSSOFLI
<o_prefix>.<o_version>.ENCRINT
<o_prefix>.<o_version>.ENCREXT
<o_prefix>.<o_version>.GTWPARM
<o_prefix>.<o_version>.ACF2GDB
```

```
<o_prefix>.<o_version>.ACF2UDB
```

3E – Encryption Datasets

The following datasets are used during the operation of Connector for ACF2 and must be copied from the source system to the target system:

```
<o_prefix>.<o_version>.ENCREXT
```

```
<o_prefix>.<o_version>.ENCRINT
```

These datasets are allocated in the previous step, **3D – Allocate Connector for ACF2 work Datasets**, and must be overwritten in this step.

4 – Adjust MVS System Parameters

4A – APF Authorized Library

The Connector LOAD and CTRANS libraries must be defined as APF authorized libraries in the target system. (for more information, see installation step 9 – Customize Communication Settings).

4B – Security Rules Definition

As part of Connector for ACF2 installation, several ACF2 definitions were setup for Connector for ACF2 datasets and started tasks. (For more information, see installation step 10 – Define Connector for ACF2 in ACF2.)

Apply the same definitions to the target system.

5 – Customize ACF2 Support

The customization process required for completing Connector for ACF2 installation is described in ACF2 Support Customization.

The process includes the installation of the SMF IEFU83 exit that is required for interception of ACF2 database security administration events.

To install support for this capability for the target system, perform the necessary instructions as described in the above chapter.

Appendix E: Managed System-Specific fields

This appendix provides reference tables for Managed System-specific fields.

Description of Table Column Titles **146**

User Fields **147**

Group Fields **159**

Description of Table Column Titles

Due to the many columns of information contained in the tables in this appendix, abbreviated column names are used. This section describes the meaning of the column titles for the Managed System-specific field tables later in the appendix.

The columns described in the following table appear in all the Managed System-specific Field tables in this appendix.

Column Title	Description	
Field	Field name (as it appears in the Details window of SailPoint). For list fields, the subfields are indented. By default, field labels are displayed in a Details window. To view field names, click the right mouse button anywhere in the Details window in the SailPoint (except on a field) and choose the option **Show Field Names** from the pop-up menu. The field names are displayed instead of the field labels.	
L	Whether or not the field accepts a list of values. A list consists of values separated by commas. Possible values in this column are:	
	L	Identifies a list field.
	S	Identifies a subfield of a list field (names of subfields are indented in the **Field** column).
T	Type of input accepted in the field. Possible values in this column are:	
	C	Character. All input is treated as characters even if all are digits.
	F	Flag. Input must be **Y** or **N**.
	N	Integer. Input must be numeric.
	T	Time. Input must be in the time format specified in the column **Restrictions**
	D	Date/Time. Input must be in the format specified in the column **Restrictions**. This format generally requires that the value be specified as a string consisting of the date or date/time.
	S	Selection from a list of predefined values.

Column Title	Description
Len	Maximum number of characters in a character field. This field length only applies if the type (column **T**) is **C**. (Field length limitations for other data types are determined by information in columns **L** and **Restrictions**.)
Restrictions	Validation restrictions such as numeric ranges or list of possible values. Underlined values (if any) are the default values.

The column titles for each type of entity differ slightly. The following table describes the meaning of the single-letter column titles used to indicate the type of function for which each Managed System-specific field is relevant.

> **Note**
> An **X** appearing in a column for a given field indicates that the field is relevant to that function.

Function Type	Column Title	Description
User	A	Add user
	U	Update user
	G	Get user
	D	Delete user
	R	Revoke/restore user
	P	Update password
Group	A	Add group
	U	Update group
	G	Get group
	D	Delete group
User–Group Connection	C	Connect user to group
	U	Update user to group connection
	G	Get user to group connection
	D	Disconnect user from group

User Fields

The table in this topic lists managed system user fields, and details additional information to consider as you configure them.

User Profile Attributes

The ACF2 application has special support USER PROFILE attributes. User Profiles are an extension of the basic User main record.

SailPoint Supported User Profile Attributes

- CICS
- DCE
- KERB
- KERBLINK
- LANGUAGE
- LINUX
- LNOTES
- NDS
- NETVIEW
- OMVS
- OPERPARM
- WORKATTR
- PASSWORD (partial support)
- PWPHRASE (partial support)

Complete List of User Profile Attributes

- CERTDATA
- CICS

- DCE
- EIM
- KERB
- KERBLINK
- KEYRING
- LANGUAGE
- LINUX
- LNOTES
- NDS
- NETVIEW
- OMVS
- OPERPARM
- PASSWORD
- PWPHRASE
- PROXY
- SECLABEL
- WORKATTR

Enabling User Profile Attributes for Configuration

Configuring user profile attributes may require you to enable a flag before you can configure and aggregate them. You can set the user profile attribute's corresponding flag by enabling it in your provisioning policy. If a user profile attribute has a corresponding flag you need to enable, it is listed in the table in the following format:

```
<user_profile_attribute_name>_SWITCH
```

For example, before you can configure any of the **WORKATTR** user profile attributes (WORKATTR.WAACCNT, WORKATTR.WABLDG, WORKATTR.WANAME, etc.), you need to set **WORKATTR_SWITCH** to a value of `Y` in your provisioning policy.

> **Note**
> This applies to any AddUser and UpdateUser provisioning policies.

Field	L	T	Len	Restrictions	M	A	U	G	D	R	P
LID		C	8			X	X	X	X	X	X
NAME		C	20			X	X	X			
groups	L	C	16K					X			
PHONE		C	12			X	X	X			
UID		C	24					X			
UIDX	L	C	24					X			
GROUP		C	8			X	X	X			
HOMENODE		C	8					X			
SEC-VIO		N	5	0-32767				X			
UPD-TOD		D	13	YYYYMMD-D HHMM				X			
MON-LOG		F	1			X	X	X			
MONITOR		F	1			X	X	X			
TRACE		F	1			X	X	X			
TSO-TRC		F	1			X	X	X			
ACTIVE		D	8	YYYYMMD-D		X	X	X			
EXPIRE		D	8	YYYYMMD-D		X	X	X			
CSDATE		D	8	YYYYMMD-D				X			
CSWHO		C	8					X			
SECURITY		F	1			X	X	X			
ACCOUNT		F	1			X	X	X			
MAINT		F	1			X	X	X			
LEADER		F	1			X	X	X			
AUDIT		F	1			X	X	X			
ACCTPRIV		F	1			X	X	X			

Field	L	T	Len	Restrictions	M	A	U	G	D	R	P
TSO		F	1			X	X	X			
JOB		F	1			X	X	X			
CICS		F	1			X	X	X			
IDMS		F	1			X	X	X			
STC		F	1			X	X	X			
CMD-PROP		F	1			X	X	X			
PRIV-CTL		F	1			X	X	X			
PPGM		F	1			X	X	X			
RSRCVLD		F	1			X	X	X			
SYNCNODE		C	8			X	X	X			
SUBAUTH		F	1			X	X	X			
TAPE-BLP		F	1			X	X	X			
TAPE-LBL		F	1			X	X	X			
CONSOLE		F	1			X	X	X			
CONSULT		F	1			X	X	X			
GRPLOGON		F	1			X	X	X			
ALLCMDS		F	1			X	X	X			
AUTODUMP		F	1			X	X	X			
DUMPAUTH		F	1			X	X	X			
BDT		F	1			X	X	X			
LOGSHIFT		F	1			X	X	X			
JOBFROM		F	1			X	X	X			
NO-INH		F	1			X	X	X			
NO-SMC		F	1			X	X	X			
NO-STORE		F	1			X	X	X			
NON-CNCL		F	1			X	X	X			
REFRESH		F	1			X	X	X			
RESTRICT		F	1			X	X	X			
RULEVLD		F	1			X	X	X			
SCPLIST		C	8			X	X	X			
PROGRAM		C	8			X	X	X			
READALL		F	1			X	X	X			
VAX		F	1					X			

Field	L	T	Len	Restrictions	M	A	U	G	D	R	P
VM		F	1			X	X	X			
VMBATCH		F	1			X	X	X			
VMBATMON		F	1			X	X	X			
VMSAF		F	1			X	X	X			
VMXA		F	1			X	X	X			
SRF		F	1			X	X	X			
VSESRF		F	1			X	X	X			
AUTOALL		F	1			X	X	X			
AUTONOPW		F	1			X	X	X			
AUTOONLY		F	1			X	X	X			
DG84DIR		F	1			X	X	X			
DIALBYP		F	1			X	X	X			
LDEV		F	1			X	X	X			
NOSPOOL		S	8	NULL | PREVENT | LOG | ALLOW		X	X	X			
SYNERR		S	8	NULL | PREVENT | LOG | ALLOW		X	X	X			
VLDVMACT		F	1			X	X	X			
ACC-CNT		N	7					X			
ACC-DATE		D	8	YYYYMMD-D				X			
ACC-SRCE		C	8					X			
ACC-TIME		C	5	HH:MM				X			
PSWD-EXP		F	1			X	X	X			
PSWD-DAT		D	8	YYYYMMD-D				X			
PSWD-INV		N	5	0-32767				X			
PSWD-SRC		C	8					X			
PSWD-TIM		C	5	HH:MM				X			
PSWD-TOD		D	13	YYYYMMD-				X			

Field	L	T	Len	Restrictions	M	A	U	G	D	R	P
				D HHMM							
PSWD-VIO		N	5	0-32767				X			
MAXDAYS		N	3	0-255		X	X	X			
MINDAYS		N	3	0-255		X	X	X			
CHAR		C	2								
DFT-DEST		C	8			X	X	X			
DFT-PFX		C	8			X	X	X			
DFT-SOUT		C	1			X	X	X			
DFT-SUBC		C	1			X	X	X			
DFT-SUBH		C	1			X	X	X			
DFT-SUBM		C	1			X	X	X			
LGN-RCVR		F	1			X	X	X			
LGN-SIZE		F	1			X	X	X			
LGN-DEST		F	1			X	X	X			
LGN-MSG		F	1			X	X	X			
LGN-PERF		F	1			X	X	X			
LGN-TIME		F	1			X	X	X			
LGN-UNIT		F	1			X	X	X			
ATTR2		C	4			X	X	X			
CMD-LONG		F	1			X	X	X			
MSGID		F	1			X	X	X			
VLD-ACCT		F	1			X	X	X			
VLD-PROC		F	1			X	X	X			
MODE		F	1			X	X	X			
OPERATOR		F	1			X	X	X			
PAUSE		F	1			X	X	X			
PROMPT		F	1			X	X	X			
MOUNT		F	1			X	X	X			
RECOVER		F	1			X	X	X			
TSOACCT		C	40			X	X	X			
TSOCMDS		C	40			X	X	X			
TSOFSCRN		F	1			X	X	X			
TSOPERF		N	3	0-255		X	X	X			

Field	L	T	Len	Restrictions	M	A	U	G	D	R	P
TSORBA		C	6	Hexa-decimal		X	X	X			
TSORGN		N	7	0-2097152		X	X	X			
TSOSIZE		N	7	0-2097152		X	X	X			
TSOTIME		N	4	0-1440		X	X	X			
TSOUNIT		C	8			X	X	X			
WTP		F	1			X	X	X			
INTERCOM		F	1			X	X	X			
JCL		F	1			X	X	X			
LINE		C	4			X	X	X			
MAIL		F	1			X	X	X			
NOTICES		F	1			X	X	X			
LGN-ACCT		C	8			X	X	X			
LGN-PROC		C	8			X	X	X			
PMT-ACCT		C	8			X	X	X			
PMT-PROC		C	8			X	X	X			
ACF2CICS		F	1			X	X	X			
CICSCL		C	6	Hexa-decimal		X	X	X			
CICSID		C	3			X	X	X			
CICSKEY		C	6	Hexa-decimal		X	X	X			
CICSKEYX		C	10	Hexa-decimal		X	X	X			
CICSPRI		N	3	0-255		X	X	X			
CICSRSL		C	6	Hexa-decimal		X	X	X			
IDLE		N	3	0-255		X	X	X			
MUSDLID		C	8			X	X	X			
IDMSPROF		C	32			X	X	X			
IDMSPRVS		N	5	0-32767		X	X	X			
MUSID		C	8			X	X	X			
MUSOPT		C	8			X	X	X			

Field	L	T	Len	Restrictions	M	A	U	G	D	R	P
MUSPGM		C	8			X	X	X			
MUSUPDT		F	1			X	X	X			
ZONE		C	3			X	X	X			
SHIFT		C	8			X	X	X			
SOURCE		C	8			X	X	X			
PREFIX		C	8			X	X	X			
VMACCT		F	1			X	X	X			
AUTHSUP1		F	1			X	X	X			
AUTHSUP2		F	1			X	X	X			
AUTHSUP3		F	1			X	X	X			
AUTHSUP4		F	1			X	X	X			
AUTHSUP5		F	1			X	X	X			
AUTHSUP6		F	1			X	X	X			
AUTHSUP7		F	1			X	X	X			
AUTHSUP8		F	1			X	X	X			
SMSINFO		C	8			X	X	X			
NORULE		F	1			X	X	X			
CICS.FORCE		C	1	n | y		X	X	X			
CICS.OPCLASS	L	C	250	1-24		X	X	X			
CICS.OPIDENT		C	3			X	X	X			
CICS.OPPRTY		N	3	1-255		X	X	X			
CICS.TIMEOUT		N	5	0-15555		X	X	X			
CICS_SWITCH		F	1			X	X	X			
DCE.AUTOLOG		C	1			X	X	X			
DCE.DCENAME		C	1023			X	X	X			
DCE.HOMECELL		C	1023			X	X	X			
DCE.HOMEUUID		C	36			X	X	X			
DCE.UUID		C	36			X	X	X			
DCE_SWITCH		F	1			X	X	X			
KERB.KERBNAME		C	240			X	X	X			
KERB.KERB_VIO		F	1			X	X	X			
KERB.MAXTKTLF		F	13			X	X	X			
KERB_SWITCH		F	1			X	X	X			

Field	L	T	Len	Restrictions	M	A	U	G	D	R	P
LANGUAGE.PRIMAR-Y		C	3			X	X	X			
LANGUAGE.SECON-DRY		C	3			X	X	X			
LANGUAGE_ SWITCH		F	1			X	X	X			
LNOTES.SNAME		C	64			X	X	X			
LNOTES_SWITCH		F	1			X	X	X			
NDS.UNAME		C	246			X	X	X			
NDS_SWITCH		F	1			X	X	X			
NETVIEW.CONSNAM-E		C	8			X	X	X			
NETVIEW.DOMAINS	L	C	90			X	X	X			
NETVIEW.IC		C	255			X	X	X			
NETVIEW.MSGRECV-R		C	1	n | y		X	X	X			
NETVIEW.NGMFAD-MN		C	1	n | y		X	X	X			
NETVIEW.NTVCLAS-S	L	C	100	y		X	X	X			
NETVIEW.SECCTL		C	7	GLOBAL | GENERAL		X	X	X			
NETVIEW_SWITCH		F	1			X	X	X			
OMVS.ASSIZE		F	13	10485760-214748364-7		X	X	X			
OMVS.CPUTIME		F	13	7-214748364-7		X	X	X			
OMVS.FILEPROC		F	5	3-65535		X	X	X			
OMVS.HOME		C	1023			X	X	X			
OMVS.MMAPAREA		F	8	1-16777216		X	X	X			
OMVS.OMVSPGM		C	1023			X	X	X			
OMVS.PROCUSER		F	5	3-32767		X	X	X			

Field	L	T	Len	Restrictions	M	A	U	G	D	R	P
OMVS.THREADS		F	6	0-100000		X	X	X			
OMVS.UID		F	13	0-214748364-7		X	X	X			
OMVS_SWITCH		F	1			X	X	X			
OPERPARM.ALTGR-OUP		C	8			X	X	X			
OPERPARM.AUTH		C	7	MASTER | CONSOLE | IO | INFO | ALL | SYSTEM		X	X	X			
OPERPARM.AUTO		C	3	no | yes		X	X	X			
OPERPARM.CMDSY-S		C	8			X	X	X			
OPERPARM.DOM		C	6	ALL | NONE | NORMAL		X	X	X			
OPERPARM.KEY		C	8			X	X	X			
OPERPARM.LOGCM-D		C	3	NO | YES		X	X	X			
OPERPARM.MFORM	L	C	40	TIME | SYSID | JOBNAME | MESSAGE | EXEMPT		X	X	X			
OPERPARM.MIGID		C	3	NO | YES		X	X	X			
OPERPARM.MONIT-OR	L	C	40	JOBNAME | JOBTIMES | TSOSESS | TSOTIME | STATUS		X	X	X			
OPERPARM.MSGLE-VEL	L	C	91	OPER_ REPLY | IMMEDIAT-E |		X	X	X			

Field	L	T	Len	Restrictions	M	A	U	G	D	R	P
				CRITICAL_ EVENTUAL | EVENTUAL | INFO | NO_ BROADCA-ST | ALL							
OPERPARM.OPERM-SCP		C	1000			X	X	X			
OPERPARM.ROUTC-ODE	L	C	250			X	X	X			
OPERPARM.STORA-GE		F	4	0-2000		X	X	X			
OPERPARM.UD		C	3	yes | no		X	X	X			
OPERPARM_ SWITCH		F	1			X	X	X			
REVOKE_TYPE		C	8	SUSPEND | CANCEL | BOTH		X	X	X		X	
WORKATTR.WAACC-NT		C	255			X	X	X			
WORKATTR.WAADD-R1		C	60			X	X	X			
WORKATTR.WAADD-R2		C	60			X	X	X			
WORKATTR.WAADD-R3		C	60			X	X	X			
WORKATTR.WAADD-R4		C	60			X	X	X			
WORKATTR.WABLD-G		C	60			X	X	X			
WORKATTR.WADEP-T		C	60			X	X	X			
WORKATTR.WANAM-		C	60			X	X	X			

Field	L	T	Len	Restrictions	M	A	U	G	D	R	P
E											
WORKATTR.WAROO-M		C	60			X	X	X			
WORKATTR_ SWITCH		F	1			X	X	X			
Expired		F	1	Hidden if the field **Support Password Reset in Change Password** is enabled in SailPoint.		X	X				X
NOExpired		F	1	Hidden if the field **Support Password Reset in Change Password** is disabled in SailPoint.		X	X				X
targetPermissions[1]		C	4096					X	X		

1. All permissions are included within targetPermissions attribute. Full description of targetPermissions content appears in Resource ACL Functions.

Group Fields

The following table lists managed system group fields.

Field	L	T	Len	Restrictions	M	A	U	G	D
UIDMASK		C	24			X	X	X	X
DESCRIPTION		C	52			X	X	X	
targetPermissions[1]		C	4096					X	X

1. All permissions are included within targetPermissions attribute. A full description of targetPermissions content appears in Resource ACL Functions.

Appendix F: Connector for ACF2 Batch Utility

The batch utility enables a user to execute a batch job containing provisioning or list requests directly on the ACF2 Connector without the requirement of a partner (such as SailPoint). The provisioning or list requests are processed by the requested Managed System Interface together with the requested security product, ACF2.

The batch utility would be helpful in the following scenarios:

- When a new installation is performed and a connection has not yet been established with SailPoint.

  Provisioning or list transactions can be issued locally within the Mainframe using the batch utility to ensure that the ACF2 Connector is installed and working properly.

- For testing or debugging purposes as instructed by SailPoint Support.

- When multiple provisioning transactions are required to be performed quickly and easily from the Mainframe than from SailPoint.

Whenever provisioning transactions are issued by the batch utility and if the Online Interceptor is not active, it is required to issue full aggregation in order to update SailPoint with the changes done by these provisioning transactions.

This appendix describes the syntax rules, provisioning and list requests, and invocation JCL required to execute the utility and an example of utility control statements with the sample job output.

Security Requirements **160**

Input Control Statements Syntax Rules **161**

Environment Definition Syntax **161**

Batch Provisioning and List Requests **162**

Invocation JCL **168**

Security Requirements

Before running the utility, ensure that the following security requirements are met:

- When invoking the batch utility with provisioning requests, the user specified as ADMIN_UNAME must have sufficient authority in order to execute the requests.

- If **STCJOBS** are used for the product started tasks, the file allocated for EXECOUT DD statement is a permanent file.

- The user submitting the job must have ALTER authority to this file.
- The user specified as ADMIN_UNAME must have UPDATE authority to this file.

  The file name is: <`olprefs`>.<`olvers`>.EXECOUT.<`procprefs`>BATCH

  where `olprefs`, `olvers` and `procprefs` are the values set in 1 – Set the Parameter Values.

Input Control Statements Syntax Rules

The following rules must be followed for the utility control statements:

- Must contain environment and provisioning / list request lines
- Data must be written in columns 1 to 72.
- The file must not have sequence numbers, that is, number mode off or unnum.
- A comment line must begin with an asterisk in column 1. The Input Control Statements can have many comment lines. Comments must not be inserted in a non-comment line.
- The input for the utility must begin with environment lines as defined in Environment Definition Syntax.
- The input for the utility may contain multiple provisioning / list requests. Each request begins with a request line which defines the request and is followed by one or more parameter lines which provides request details.
- The input control statements – keywords and values – would not be converted to upper or lowercase. Hence all data must be specified in the correct case.
- If a value is longer than 1 line, it must be surrounded by parentheses and continued in column 1 of the next line. For example,

```
Keyword=(value1,value2,value3,value4,value5,value6,
value7,value8,value9,value10,value11,value12,value13)
```

Environment Definition Syntax

The utility input must begin with environment definition lines.

```
:ENV
```

Parameter	Mandatory/Optional	Description
RSS_TYPE	Mandatory	ACF2
RSS_NAME	Mandatory	The name defined for RSSNAME in the DEFPARMS member

Parameter	Mandatory/Optional	Description
ADMIN_UNAME	Mandatory	The administrator User ID
ADMIN_GROUP	Optional	The group to be used for the administrator. **Note** Used to override the default group used during logon processing.

Batch Provisioning and List Requests

This section describes the following provisioning and list requests:

- ADDUSER – Add a User
- UPDUSER – Update a User
- DELUSER – Delete a User
- DISABLEUSER – Disable a User
- ENABLEUSER – Enable a User
- LISTUSER – List User Information
- CHGPWD – Change a User password
- ADDGROUP – Add a Group
- UPDGROUP – Update a Group
- DELGROUP – Delete a Group
- LISTGROUP – List Group Information
- ADDCONN – Add a User-Group Connection

- DELCONN – Delete a User-Group Connection
- LISTCONN – List User-Group Connection information
- LISTACL – List ACL permission information

ADDUSER – Add a User

Use these request and parameter lines to add a user. To set user privileges, use the optional AUDIT=Y, SECURITY=Y, ACCOUNT=Y, CONSULT=Y, and LEADER=Y keyword lines.

```
:ADDUSER=userid
```

Parameter	Mandatory/Optional	Description/Value
USER.PASSWORD	Mandatory	Password of the user. For example: `USER.PASSWORD=[(]password[,PERM / TEMP)]`
USER.DISABLE	Optional	`USER.DISABLE=YES`

kwd	Optional	The kwd lines represent attributes which are specified in the SailPoint schema.
		`kwd=value`
		or
		`kwd=(value1,value2,value3,value4...)`

UPDUSER – Update a User

Use these request and parameter lines to update a user. To set user privileges, use the optional AUDIT=Y, SECURITY=Y, ACCOUNT=Y, CONSULT=Y, and LEADER=Y keyword lines.

```
:UPDUSER=userid
```

Parameter	Mandatory/Optional	Description
USER.PASSWORD	Optional	`USER.PASSWORD=[(]password[,PERM / TEMP)]`
USER.DISABLE	Optional	`USER.DISABLE=YES`
USER.ENABLE	Optional	`USER.ENABLE=YES`
kwd	Optional	The kwd lines represent attributes which are specified in the SailPoint schema. `kwd=value`

Parameter	Mandatory/Optional	Description
		or `kwd=(value1,value2,value3,value4...)` To nullify a keyword, the keyword must be specified with a null value. To delete one value from a multi-value keyword field, specify the keyword with all remaining values.

DELUSER – Delete a User

Use these request and parameter lines to delete a user.

```
:DELUSER=userid
```

Parameter	Mandatory/Optional	Description
NORULE	Optional	Avoids deletion of user dataset rules. NORULE=Y

DISABLEUSER – Disable a User

Use this request line to disable a user.

```
:DISABLEUSER=userid
```

Parameter	Mandatory/Optional	Description
REVOKE_TYPE	Mandatory	REVOKE_TYPE=SUSPEND / CANCEL / BOTH

ENABLEUSER – Enable a User

Use this request line to enable a user.

```
:ENABLEUSER=userid
```

Parameter	Mandatory/Optional	Description
REVOKE_TYPE	Mandatory	REVOKE_TYPE=SUSPEND / CANCEL / BOTH

LISTUSER – List User Information

Use these request and parameter lines to list user information.

```
:LISTUSER
```

Default parameter setting:

```
FILTER.USERID=*
```

Parameter	Mandatory/Optional	Description
FILTER.USERID	Optional	`FILTER.USERID={* | [()]userid[,userid…]())] | userid-mask}`
USER.GETCONN	Optional	Indicates that the groups keyword would contain all the groups associated with the user. `USER.GETCONN=NO/YES`
kwd	Optional	The specified kwd lines are attributes which are retrieved and displayed. If no kwd is specified all user related attributes are retrieved and displayed. `kwd`

CHGPWD – Change a User Password

Use these request and parameter lines to change a user's password.

```
:CHGPWD=userid
```

Parameter	Mandatory/Optional	Description
USER.PASSWORD	Optional	User password `USER.PASSWORD =[()]password [,PERM / TEMP)]`
EXPIRED	Optional	Overrides the password life (perm / temp) `EXPIRED=Y`
NOEXPIRED	Optional	Overrides the password life sub-parameter and the EXPIRED parameter. `NOEXPIRED=Y`
VERIFY_PWD	Optional	Indicates that the password is to be verified, not changed. `VERIFY_PWD=N / Y`

ADDGROUP – Add a Group

Use these request and parameter lines to add a group.

```
:ADDGROUP=groupid
```

Parameter	Mandatory/Optional	Description
GROUP.PARENT	Optional	`GROUP.PARENT =parent-group-name`

kwd	Optional	The kwd lines represent attributes which are specified in the SailPoint schema.
		`kwd=value`
		or
		`kwd=(value1,value2,value3,value4...)`

UPDGROUP – Update a Group

Use these request and parameter lines to update a group.

```
:UPDGROUP=groupid
```

Parameter	Mandatory/Optional	Description
GROUP.PARENT	Optional	`GROUP.PARENT =parent-group-name`

kwd	Optional	The kwd lines represent attributes which are specified in the SailPoint schema.
		`kwd=value`
		or
		`kwd=(value1,value2,value3,value4...)`
		To nullify a keyword, the keyword must be specified with a null value.
		To delete one value from a multi-value keyword field, specify the keyword with all remaining values.

DELGROUP – Delete a Group

Use these request and parameter lines to delete a group.

```
:DELGROUP=groupid
```

LISTGROUP – List Group Information

Use these request and parameter lines to list group of information.

```
:LISTGROUP
```

Default parameter setting:

```
FILTER.GROUPID=*
```

Parameter	Mandatory/Optional	Description
FILTER.GROUPID	Optional	`FILTER.GROUPID={*|[()]groupid[,groupid…][)]}`
kwd	Optional	The specified kwd lines are attributes which are retrieved and displayed. If no kwd is specified all group-related attributes are retrieved and displayed. `kwd`

ADDCONN – Add a User-Group Connection

Use these request and parameter lines to add a user-group connection.

```
:ADDCONN=(userid, groupid)
```

Parameter	Mandatory/Optional	Description
CONN.AUTH	Optional	`CONN.AUTH=REG/ADMIN/AUDIT/ADMINAUDIT`

kwd	Optional	The kwd lines represent attributes which are specified in the SailPoint schema.
		`kwd=value`
		or
		`kwd=(value1,value2,value3,value4...)`

DELCONN – Delete a User-Group Connection

Use these request and parameter lines to delete a user-group connection.

```
:DELCONN=(userid, groupid)
```

Parameter	Mandatory/Optional	Description
CONN.AUTH	Optional	`CONN.AUTH=REG/ADMIN/AUDIT/ADMINAUDIT`
OWNER	Optional	`OWNER=connection-owner`

LISTCONN – List User-Group Connection Information

Use these request and parameter lines to list user-group connection information.

```
:LISTCONN
```

Default parameter setting:

```
FILTER.CONN=*
```

All the parameters listed below are mutually exclusive.

Parameter	Mandatory/Optional	Description
FILTER.CONN	Optional	All user-group connections or a specific user-group connection `FILTER.CONN={* | (userid,groupid)}`
FILTER.GROUP	Optional	All user connections with the specific group(s) `FILTER.GROUP=[(]groupid[,groupid…)]`
FILTER.USER	Optional	All group connections with the specific user(s) `FILTER.USER=[(](userid[,userid…)]`

LISTACL – List ACL (Permission) Information

Use these request and parameter lines to list ACL (permission) information

```
:LISTACL
```

Parameter	Mandatory/Optional	Description
FILTER.RESTYPE	Mandatory	FILTER.RESTYPE=resource-type
FILTER.RESNAME	Mandatory	FILTER.RESNAME=resource-name
kwd	Optional	The specified kwd lines are attributes which are retrieved and displayed. If no kwd is specified, all permission-related attributes are retrieved and displayed. `kwd=`

Invocation JCL

> **Note**
> Before running the utility, refer to the Security Requirements.

Use the CTSBATCH JCL member to invoke the batch utility. This job calls the batch utility from the CS Server procedure but uses a different program name (EXEC CTSACS,PROG=CTSCBAT). The input control statements for the utility are provided in DD statement SYSIN.

To use a different input file name, specify the TOKEN parameter and add the DD statement for the input file as follows:

```
//CALLCBAT EXEC CTSACS,PROG=CTSCBAT,TOKEN=(DDIN(BATINPUT))
//CTSACS.BATINPUT DD DSN=SPIIQ.V4000.INPUT(BATCHFAC),DISP=SHR
```

SYSIN File Example

The following is an example of a SYSIN file specifying several provisioning /list requests:

```
RSS_TYPE=ACF2
:ENV
RSS_NAME=MVSACF2
ADMIN_UNAME=SECAL
*****************
:CHGPWD=SECST1
USER.PASSWORD=aBc@DeF
*****************
:ENABLEUSER=SECST1
REVOKE_TYPE=SUSPEND
*****************
:LISTUSER
FILTER.USERID=(SECALT,SECSTT,SECST1,SECSTR)
USER.GETCONN=YES
UID
UIDX
NAME
PREFIX
GROUP
TSOPROC
ACCTPRIV
PSWD_SRC
PSWD_TOD
PSWD_DAT
groups
```

Sample Batch Job Output

When executing the batch utility with the provisioning/ list requests as displayed in the SYSIN input file above, the following output is printed in the SYSPRINT file:

```
*** Control-SA Batch Program ***
================================
Control-SA 4.0.00 - BS10062
================================
:CHGPWD:
 User     : SECST1
 Password : aBc@DeF
 Password
  life     : Ignored
  ADDINFO: TYPE    KEYWORD/VALUE
API call :CHGPWD OK
:ENABLEUSER:
 User    : SECST1
 Password: \
API call :ENABLEUSER OK
:LISTUSER:
User List
=========
User: SECALT
  Group    :
```

```
  Password :
  Status   : Normal
  Authority: Administrator
  Password
     status: Ignored
  ADDINFO: TYPE    KEYWORD/VALUE
           1A:     UID = A1234567BXYZXW  SECALT
           1A:     UIDX = A1234567BXYZXW  SECALT
           1A:     NAME = IUIU
           1A:     PREFIX = SECALT
           1A:     GROUP =
           1A:     TSOPROC = LOGONSEC
           1A:     ACCTPRIV = Y
           1A:     PSWD_SRC = SEC60005
           1A:     PSWD_TOD = 20161006025543
           1A:     PSWD_DAT =
           1B:     groups = ********B
User: SECSTT
  Group    :
  Password :
  Status   : Normal
  Authority: Regular
  Password
     status: Ignored
  ADDINFO: TYPE    KEYWORD/VALUE
           1A:     UID =         A       SECSTT
           1A:     UIDX =
           1A:     NAME = ZIGGY
           1A:     PREFIX = SECSTT
           1A:     GROUP =
           1A:     TSOPROC =
           1A:     ACCTPRIV = N
           1A:     PSWD_SRC =
           1A:     PSWD_TOD = 20170503034349
           1A:     PSWD_DAT =
           1B:     groups = ********A
User: SECST1
  Group    :
  Password :
  Status   : Normal
  Authority: Regular
  Password
     status: Ignored
  ADDINFO: TYPE    KEYWORD/VALUE
           1A:     UID =         C       SECST1
           1A:     UIDX =
           1A:     NAME = ZIGGY
           1A:     PREFIX = SECST1
           1A:     GROUP =
           1A:     TSOPROC =
           1A:     ACCTPRIV = N
           1A:     PSWD_SRC =
           1A:     PSWD_TOD = 20170918012406
           1A:     PSWD_DAT =
```

```
           1B:     groups = ********C
User: SECSTR
  Group    :
  Password :
  Status   : Normal
  Authority: Regular
  Password
     status: Ignored
  ADDINFO: TYPE    KEYWORD/VALUE
           1A:     UID =          C       SECSTR
           1A:     UIDX =
           1A:     NAME =
           1A:     PREFIX = SECSTR
           1A:     GROUP =
           1A:     TSOPROC =
           1A:     ACCTPRIV = N
           1A:     PSWD_SRC = SEC60004
           1A:     PSWD_TOD = 20170910064616
           1A:     PSWD_DAT =
           1B:     groups = ********C
*** Total number of users found: 4 ***
```