Redbooks
ibm.com/redbooks

IBM MFA V1R1
TouchToken, PassTicket, and Application Bypass Support

Keith Winnard
John Petreshock
Philippe Richard

Security

z Systems

IBM

Redpaper

International Technical Support Organization

IBM MFA V1R1: TouchToken, PassTicket, and Application Bypass Support

December 2016

REDP-5386-00

Note: Before using this information and the product it supports, read the information in “Notices” on page v.

First Edition (December 2016)

This edition applies to Version 1 Release 1 of IBM Multi-Factor Authentication for z/OS (product number 5655-162).

This document was created or updated on December 22, 2016.

Contents

Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, MD-NC119, Armonk, NY 10504-1785, US

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at “Copyright and trademark information” at http://www.ibm.com/legal/copytrade.shtml

The following terms are trademarks or registered trademarks of International Business Machines Corporation, and might also be trademarks or registered trademarks in other countries.

IBM®	Redbooks®	z/OS®
IBM z Systems®	Redpaper™	z13™
IBM z13®	Redbooks (logo)  ®	zEnterprise®
MVS™	VTAM®	
RACF®	z Systems®	

The following terms are trademarks of other companies:

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

Preface

What is IBM® Multi-Factor Authentication (MFA)? IBM MFA consists of the following elements:

- Something that you know, such as a Personal Identification Number (PIN) or a password.
- Something that you are, such as a finger print or retinal scan.
- Something that you have, such as a hard token (for example, a key fob or soft token), which is software-based).

This IBM Redpaper™ publication helps you install, customize, and configure IBM MFA for z/OS® V1.1.0. It also provides information that is based on our experience in a controlled environment and includes the following chapters:

- Chapter 1, “IBM MFA Overview” on page 1: Describes product information, authentication factors, and IBM RACF® support for IBM MFA for z/OS.
- Chapter 2, “MFA installation and basic customization” on page 13: Describes the basic installation and initial configuration work.
- Chapter 3, “Preparing for IBM TouchToken” on page 29: Describes basic configuration work, how to prepare ICSF, setting up the PKCS#11 token, and configuring the AT-TLS profile.
- Chapter 4, “Configuring IBM TouchToken” on page 63: Introduces Time-based One-Time Password (TOTP), and basic installation work on the Apple iOS device.
- Chapter 5, “User account administration and logon” on page 77: Provides examples of setting up users and pass codes and logging on to multiple applications.
- Chapter 6, “MFA application selection and PassTicket support” on page 101: Describes how applications can be bypassed for MFA processing. Support for PassTickets was added in this release of the software.
- Chapter 7, “Operational information and FAQs” on page 113: Provides high-level operational information and a FAQ section that addresses practical situations that can arise.
- Appendix A, “Sample AT-TLS policy for use with MFA” on page 119: Presents a sample AT-TLS policy for use with MFA.

Authors

This paper was produced by a team of specialists from around the world working at the International Technical Support Organization, Poughkeepsie Center.

Keith Winnard is a z/OS Project Leader at the International Technical Support Organization, Poughkeepsie Center. He writes extensively and is keen to engage with customers to understand what they want from IBM Redbooks® publications. Before joining the ITSO in 2014, Keith worked for clients and Business Partners in the UK and Europe in various technical and account management roles. He is experienced with blending and integrating new technologies into the traditional role of mainframes.

John Petreshock is an IBM z Systems® Security Offering Manager, PMP, in the IBM Systems Group based in Poughkeepsie, NY. He has been with IBM since 1997 and has had roles in software development, test, management, and is a PMP-certified project manager as an offering manager in the IBM Systems Group.

Philippe Richard works in the IBM client Center in Montpellier, France. He joined IBM France in 1985 to work in software support for IBM MVS™. Philippe has held several positions, including teaching, systems programming and consultancy on Migration, education, and project planning. Philippe is now the worldwide lead developer of technical training and classes for STG IBM z Systems™ where he manages the z/OS curriculum content, including z/OS , RACF, Parallel Sysplex/UNIX System Services/WebSphere, and Liberty for z/OS. He has contributed to other Redbooks publication projects and is a regular speaker at IBM conferences in Europe (for example, STG university and Security conference), and customers Guide/Share security meetings.

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our papers to be as helpful as possible. Send us your comments about this paper or other IBM Redbooks publications in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

  ibm.com/redbooks

- Send your comments in an email to:

  redbooks@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- Find us on Facebook:

  http://www.facebook.com/IBMRedbooks

- Follow us on Twitter:

  http://twitter.com/ibmredbooks

- Look for us on LinkedIn:

  http://www.linkedin.com/groups?home=&gid=2130806

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm

- Stay current on recent Redbooks publications with RSS Feeds:

  http://www.redbooks.ibm.com/rss.html

1

IBM MFA Overview

This chapter provides basic product information for IBM Multi-Factor Authentication (MFA) for z/OS software product and an overview of multi-factor authentication.

This chapter includes the following topics:

1.1 IBM MFA for z/OS Overview

In this section, we describe the IBM MFA for z/OS software product.

1.1.1 Product information

The software product includes the following product numbers:

- 5655-162: IBM Multi-Factor Authentication for z/OS V1.1.0
- 5655-163: IBM Multi-Factor Authentication for z/OS S&S V1.1.0

IBM MFA for z/OS features the following requirements:

- z/OS V2.1 with z/OS Security Server with PTFs for APAR OA48359 or z/OS V2.2 with z/OS Security Server with PTFs for APAR OA48359
- RSA Authentication Manager 8.1 or later for RSA SecurID exploitation

The z/OS continuous delivery model features the following key functionality release dates:

- MFA V1.0 was generally available first quarter 2016
- MFA Service Stream Enhancements released in second quarter 2016 (APAR OA50016):
  - IBM TouchToken support
  - Application selection
  - RACF PassTicket support
- IBM TouchToken application for iOS release in August 2016

Documentation

The following documentation is available to help you install, customize, and integrate IBM MFA for z/OS into your z/OS environment:

- For installation information, refer to *IBM Multi-Factor Authentication for z/OS Program Directory,* which is included in the product package.
- *IBM Multi-Factor Authentication for z/OS Installation and Customization,* SC27-8447-01
- *IBM Multi-Factor Authentication for z/OS User's Guide*, SC27-8448-01

The following RACF publications include changes to support IBM MFA for z/OS with APAR OA50016:

- *z/OS Security Server RACF Security Administrator's Guide*, SA23-2289
- *z/OS Security Server RACF Callable Services,* SA23-2293
- *z/OS Security Server RACF Macros and Interfaces* SA23-2288
- *z/OS Security Server RACF Messages and Codes* SA23-229100
- *z/OS Security Server RACF Command Language Reference,* SA23-2292
- *z/OS Security Server RACF Auditor's Guide,* SA23-2290
- *z/OS Security Server RACROUTE Macro Reference*, SA23-2294
- *z/OS Security Server RACF Data Areas,* GA32-0885
- *z/OS Security Server RACF General User's Guide,* SA23-2298

1.2 Multi-factor authentication

Multi-factor authentication is described in this section.

1.2.1 Authentication factors

The following authentication factors are featured:

- Something you know, such as a Personal Identification Number (PIN) or a password
- Something you are, such as a finger print or retinal scan
- Something you have, such as a hard token (for example, a key fob) or soft token, which is software-based

A multi-factor authentication system requires that multiple authentication factors be presented during logon to verify a user's identity. Each authentication factor must be from a separate category of credential types.

How IBM MFA for z/OS helps

IBM MFA for z/OS provides a way to raise the assurance level of OS and applications and hosting environments by extending RACF to authenticate users with multiple authentication factors. It integrates with RACF in the following ways:

- Allows RACF to use RSA SecurID and Apple Touch ID device authentication mechanisms in place of the standard z/OS password.
- Tightly integrated with SAF and RACF:
  - RACF provides the configuration point to describe multi-factor authentication requirements down to a per-user ID basis.
  - Configuring and provisioning data that is stored in RACF database allows seamless back-up and recovery.
- Provides various approaches to implement IBM MFA natively on z/OS. Multiple tokens are supported, which allows clients to choose the factors that suit local security policy.
- Supports two methods of MFA:
  - Applications that were modified to support passing a second factor within the authentication dialogue of the application or the password or phrase field are "repurposed" to permit the specification of the second factor if the application was not modified to support MFA (also supports non-biometric token types).
  - Applications that were not modified to support passing a second factor within the authentication dialogue. Users log on with their z/OS user ID and password or phrase. Biometrics are supported through this authentication channel.

1.2.2 RSA authentication manager

IBM MFA for z/OS can support the RSA authentication manager concepts. Three concepts are briefly described in this section.

RSA SecurID token code

A token code is a continuously regenerated number that is used to prove your identity. The token code is a pseudo-random 6- or 8-digit number (PRN) that is based on the current time and is displayed on the RSA SecurID token device. It is presumed that only an authorized user possesses the token device.

The token code is a one-time password (OTP). It is valid only when it is displayed and it can be used only once. The token device generates a new token code at regular intervals (typically, every 60 seconds).

RSA SecurID PIN

The SecurID PIN is conceptually similar to a PIN that you might use for financial transactions. It is a number that only you know and helps to identify you. It is a unique 4- to 8-digit identifier that only you should know. Your PIN can be of your own choosing, or system generated by RSA Authentication Manager depending on your RSA token policy.

If you create your own PIN, follow the locally established rules for creating a valid PIN, such as number of characters and the reuse policy.

Your security administrator can clear and reset the PIN as needed, so it is possible that your current PIN becomes invalid and you must change it.

RSA SecurID passcode

A SecurID passcode is a combination of a PIN and token code. Similar to the token code, a passcode is an OTP. It is valid only when it is displayed and it can be used only once. The following types of passcodes are available:

- For hardware fob-style tokens without a PINpad, the SecurID passcode consists of your PIN followed by the token code (you must enter both). For example, if your PIN is 1234 and the token code is 567891, you enter the passcode as 1234567891.
- For SecurID PINpad hardware tokens and SoftToken applications, you enter your PIN on the pin pad and the token generates a hash-encrypted passcode from the PIN and the generated token. The token generates a new passcode at regular intervals (often every 60 seconds). You use the generated passcode when you log in.

Note: In the first type of passcode, the PIN *and* token code are specified for passcode. In the second type of passcode, the PIN is used by the token to generate a passcode.

1.2.3 Types of token devices

In this section, we describe the three types of token devices.

RSA SecurID card style tokens and key fobs

RSA SecurID card style tokens and key fobs generate a token code. Card-style tokens (such as the RSA SecurID 200) and key fobs (such as the RSA SecurID 800) function identically, with both displaying the token code in the LCD, as shown in Figure 1-1.

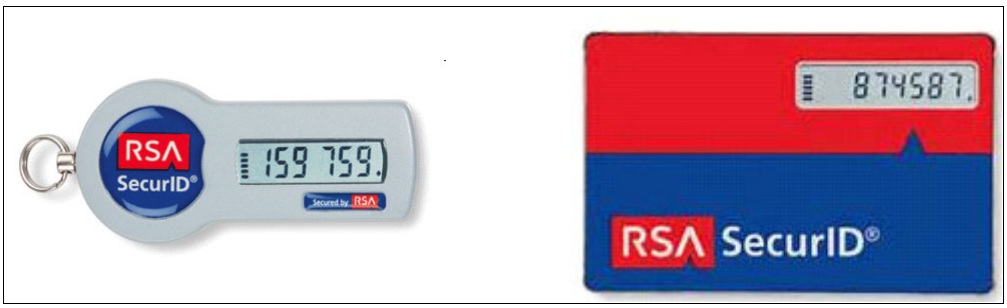

Figure 1-1 RSA SecurID card style tokens and key fobs

RSA SecurID PINpads

With an RSA SecurID PINpad token, you enter your PIN directly into the token and the token generates a hash-encrypted six- or eight-digit passcode, as shown in Figure 1-2. You can use the PINpad token in the following ways:

- If you have a valid PIN, enter the PIN and the token generates a hash-encrypted passcode. The passcode that is displayed is a hash-encrypted combination of the PIN and the current token code.
- If you do not have a valid PIN (which can occur if the security administration policies force you to change it), use the token to generate a token code. You then use the generated token code to log in and change your PIN.

Figure 1-2 RSA SecurID PINpads

RSA SecurID SoftToken tokens

RSA SecurID SoftToken applications are stored on a computer or other smart device. Next, we describe how to use the SoftToken application.

If you have a valid PIN, enter the PIN and the token generates a hash-encrypted passcode. The passcode that is displayed is a hash-encrypted combination of the PIN and the current token code. The passcode can be entered into the password field, as shown in Figure 1-3.

Figure 1-3 RSA SecurID SoftToken logon

If you do not have a valid PIN (which can occur if the security administrator forces you to change it), use the token to generate a token code. You then use the generated token code to log in and change your PIN.

The same type of token that is used for logging in to z/OSMF is shown in Figure 1-4.

Figure 1-4 Logging on to z/OSMF with a SoftToken

The user's PIN is not entered during logon processing; instead, the PIN is entered on the SoftToken software application to create the token.

Note: The MVS operator console logon does not support hard tokens. However, the MVS operator console logon supports soft tokens.

Using the password field in an application

If the application includes passphrase support, the password can be greater than 8 characters. Without passphrase support, the application has a *maximum* of 8 characters.

Example 1-1 shows a way to use the password field for an application with and without pass phrase support.

Example 1-1 Using the password field

```
Assume the following values:
PIN = 1234
Token = 567891
Soft token = 433866

Logon with passphrase support
For a hard token, you enter 1234567891 in the password field.

For a soft token, you enter 433866 in the password field.

Without passphrase support
For a hard token, you enter 567891 in the password field AND you also enter 1234
in the new password field.
(Note: You may need to enter 1234 again in the validate change field.)

For a soft token, you enter 433866 in the password field.
```

1.3 RACF support for IBM MFA for z/OS

RACF was enhanced to support IBM MFA Version 1.1 for z/OS.

1.3.1 RACF enhancements

The following enhancements were made:

- The RACF ALTUSER command allows the provisioning and definition of the acceptable MFA tokens for a user.
- ISPF panels support command extensions for MFA.
- A new SAF service is provided for z/OS MFA Services, which allows access to MFA data that is stored in the RACF database.
- Auditing functionality was added that monitors the factors that are used during the authentication process for a user.
- The RACF database unloads non-sensitive fields that were added to the RACF database that is used by MFA processing.

1.3.2 Authentication factor stored in RACF profiles

The RACF database serves as the data repository for MFA data. Consider the following points:

- MFA data is accessed by using RACF commands and a SAF/RACF callable service.
- MFA User Specific Data contains general MFA user policy information and factor-specific data for the user.
- Authentication Factor Definition defines an authentication factor and contains the following factor configuration used by MFA Services:
  - New RACF general resource class: MFADEF
  - Profile naming conventions: FACTOR.<factorName>
- Extended factor-specific data:
  - Contains extended factor-specific information for a specific user.
  - These profiles are not administered by RACF commands; rather, they are used by MFA. Services are similar to profiles in the RACF **DIGTCERT** general resource class.
  - Profiles in this class are used for large binary blobs, such as biometric data, that can be too large for the user profile. (Profile is **USER.<factorName>.<userid>**).

1.3.3 Managing MFA RACF user profiles

MFA Factor fields are stored in the RACF user profile and are defined by a RACF Administrator by using the **ALTUSER** command, as shown in Figure 1-5.

```
[ MFA (
      [ PWFALLBACK | NOPWFALLBACK ]
      [ FACTOR (factor-name) | DELFACTOR (factor-name)
     ]
      [ ACTIVE | NONACTIVE ]
      [ TAGS (tag-name:value …) ]
        | DELTAGS (tag-name …)
        | NOTAGS ]
)
| NOMFA  ]
```

Figure 1-5 ALTUSER command syntax

The information is stored in the base segment of the user's profile. Consider the following points:

- MFA cannot be specified for a PROTECTED user.
- The MFADEF class must be active before a user can log on with IBM MFA.

ALTUSER command keyword and options

The following **ALTUSER** (abbreviation ALU) command keywords are available to help you control and manage settings for users and their factor authentication:

- MFA | MOMFA

  `MFA`: Specifies multi-factor authentication information for the user profile that is changed.

- PWFALLBACK | NOPWFALLBACK:
  - `PWFALLBACK`: If IBM MFA for z/OS is unavailable, cannot determine the validity of an ACTIVE factor, or the user misplaced their token device, PWFALLBACK allows to logon to the system by using any RACF authenticators, such as their password, password phrase, or PassTicket.
  - `NOPWFALLBACK`: If IBM MFA for z/OS is unavailable or cannot determine the validity of an ACTIVE factor, this user cannot log on to the system with any RACF authenticators.
  - `NOPWFALLBACK` is the default.

- FACTOR | DELFACTOR

  `FACTOR(factor-name)`: Specifies an authentication factor for a user. If the user is not registered to the factor, the factor is added to the user. The specified factor must be defined in an MFADEF class profile named `FACTOR.<factorName>`. Other ALTUSER keywords, such as ACTIVE and TAGS, are specific to this specified factor. Consider the following points:

  - `Factor-name` is a 1 - 20 character identifier. The characters can be alphabetic, numeric, or national.
  - A user is limited to 10 factors.
  - Only one factor can be specified in a single **ALTUSER** command.

  `DELFACTOR(factor-name)`: Deletes the specified factor from the list of authentication factors registered to the user.

- ACTIVE | NOACTIVE:
  - `ACTIVE`: The user is required to authenticate to IBM MFA with the specified factor to log on to the system when the MFADEF class is active.
  - `NOACTIVE`: The user is not required to authenticate to IBM MFA with the specified factor to log on to the system.
  - `NOACTIVE` is the default.
- TAGS | DELTAGS | NOTAGS
  - `TAGS(tag-name:tag-value...)`: Specifies tags and values for the specified factor. The tag-name and tag-value pairs are factor-specific and are defined by IBM MFA for z/OS.

    ALTUSER calls IBM MFA for z/OS address space to validate `tag-names` and `tag-values`. IBM MFA must be available for RACF to process the `TAGS` keyword. IBM MFA for z/OS might reject a `tag-name` or `tag-value` during **ALTUSER** processing.

    IBM MFA for z/OS might use these values during logon processing to authenticate a user. For more information about each factor's configuration data parameters, see the MFA User guide.

    When the tag-name is not in the `TAGS` for the specified factor, the `tag-name` is added. When the `tag-name` is present for the specified factor, it is replaced with the new `tag-value`. Consider the following points:

    - The `tag-name` is a 1 - 20 character case-insensitive identifier and can consist of alphabetic or numeric characters.
    - A factor is limited to 20 tags.
    - The `tag-value` can be 1 - 1024 characters and can consist of any character. If the `tag-value` you specify includes any blanks, the `tag-name:tag-value` pair must be enclosed in quotation marks.
  - `DELTAGS(tag-name ...)`: Deletes specific tags for the specified factor. The `tag-name` is ignored when it does not exist for a specified factor.
  - `NOTAGS`: Removes all tags for the specified factor.
  - `NOMFA`: Specifies that RACF delete all MFA fields from the user's profile. The user is no longer required to provide more authentication factors when they log on.

LISTUSER command keyword and options

The MFA attributes can be listed for a use by using the **LISTUSER** (LU) RACF command. The following format is used for the command:

```
LU username MFA
```

Output from the **ALTUSER** and **LISTUSER** commands is shown in Example 1-2.

Example 1-2 ALTUSER and LISTUSER command

```
ALTUSER USERABC MFA(PWFALLBACK(Y))
ALTUSER USERABC MFA(FACTOR(RSASecurID) TAGS(SIDUSERID:RSAABC))
LISTUSER USERABC MFA
...
MFA INFORMATION:
----------------------------
PASSWORD FALLBACK IS ALLOWED.

FACTORS:
FACTOR = RSASecurID
```

```
STATUS = ACTIVE
FACTOR TAGS:
   SIDUSERID:RSAABC
```

MFA checking can be disabled for all users by deactivating the MDADEF class, as shown in the following example:

```
SETROPTS NOCLASSACT(MFADEF)
```

z/OS MFA services started task

z/OS MFA address space tracks states for user authentication events. It also provides an anchor for communications for factors, such as RSA SecurID Extensible architecture, to enable support for more authentication factors.

1.3.4 Password fallback option

PWFALLBACK is an option you define in the MFA user segment to configure the password fallback for the user. If you configure user accounts with the password fallback parameter, users can log in by using their z/OS password if the RSA Authentication Manager or IBM MFA server are not available. The password fallback mechanism is provided as a fail safe authentication method. If you omit this parameter, the default is NOPWFALLBACK.

This option is defined by using one of the two **AlterUser** (**ALU**) commands that are shown in Example 1-3.

Example 1-3 Setting the password fallback option for a user

```
ALU [Login ID] MFA(FACTOR(AZFTOTP1) ACTIVE PWFALLBACK)
ALU [Login ID] MFA(FACTOR(AZFTOTP1) NOPWFALLBACK)
```

Password fallback scenario

The following User IDs are used in this scenario:

- MFAUSE1: Includes the password fallback option
- MFAUSE2: Does not have include the password fallback option

The MFA segment for each user is shown in Example 1-4.

Example 1-4 Comparison of MFA segments

```
MFAUSE1
MULTIFACTOR AUTHENTICATION INFORMATION:
 -------------------------------------
   PASSWORD FALLBACK IS ALLOWED
   FACTOR = AZFTOTP1
     STATUS = ACTIVE
     FACTOR TAGS =
       REGSTATE:PROVISIONED
       KEYLABEL:AZF.MFASET1.D15FD378FC229063
       ALG:SHA256
       CVALUE:24592795
       NUMDIGITS:8
       PERIOD:60
       WINDOW:10
   FACTOR = AZFPTKT1
     STATUS = ACTIVE
```

```
     FACTOR TAGS =
       MFAFIRST:N

MFAUSE2
MULTIFACTOR AUTHENTICATION INFORMATION:
---------------------------------------
  PASSWORD FALLBACK IS NOT ALLOWED
  FACTOR = AZFTOTP1
    STATUS = ACTIVE
FACTOR TAGS =
       REGSTATE:PROVISIONED
       KEYLABEL:AZF.MFASET2.D171603F29653359
       ALG:SHA256
       CVALUE:24611964
       NUMDIGITS:8
       PERIOD:60
       WINDOW:10
```

The MFA server AZF#IN00 is now shut down. User MFAUSE1 can no longer enter TOTP passcodes because the server is not available.

If MFAUSE1 attempts to log on a user to any application that uses the RACF password, the attempt is successful. MFASET1's logon works for any application, even those that were not allowed with a bypass profile because the server is down and password fallback is allowed.

If the MFA server AZF#IN00 is restarted, the MFA process resumes as specified by the MFA policy. MFAUSE1 cannot logon by providing their RACF password.

If MFSUSE2 attempts to log on to any application by using their RACF password, the attempt fails because MFAUSE2 has no password fallback. If MFAUSE2 attempts to log on to TSO, the messages that are shown in Example 1-5 appear.

Example 1-5 MFAUSE2 fails to log on to TSO

```
TSO panel message:
IKJ56421I PASSWORD NOT AUTHORIZED FOR USERID

SYSLOG messages:
ICH408I USER(DRICHAR ) GROUP(OMVSGRP ) NAME(DAVID SMITH )
  LOGON/JOB INITIATION - MULTIFACTOR AUTHENTICATION UNAVAILABLE
```

2

MFA installation and basic customization

In this chapter, we describe our installation of IBM Multi-Factor Authentication (MFA) in a controlled environment. We describe the base installation procedure that is outlined in the program directory and changes to the installation process that were introduced through the maintenance stream.

This chapter includes the following topics:

2.1 Installing MFA V1.1.0

The MFA product is installed by completing the following overall steps:

1. Order and download the product by using your normal procedures, preferably with Shopz.
2. Review the program directory for the installations instructions.
3. Perform the SMP/E Receive, Apply, Accept functions.

Note: Installation changes were introduced with PTF UI40279. If you follow the program directory GI13-4316-00, you miss some important changes that are not yet reflected in the program directory.

Consider the following points:

- The MFA for z/OS product is closely integrated with z/OS Security Server RACF and centralizing authentication factor information in the RACF database.
- MFA for z/OS relies on the RACF Security Administrator to identify, which users are subject to requiring MFA policy.
- MFA is designed to work with IBM z/OS Security Server RACF to centralize the information of valid factors within RACF.
- The Security Administrator defines what authentication factors are valid for a specific z/OS instance and has control on a granular z/OS User ID level as to what users are subject to require stronger authentication.
- The z/OS Security Server RACF enablement for MFA for z/OS consists of updates to the RACF database, RACF commands, callable services, logon processing, and RACF utilities.

2.1.1 Target system operational prerequisites

The SMP/E FMID for MFA V1.1.0 is HMFA110. Although it includes no conditional installation prerequisites, the target system requires the operational prerequisites that are listed in Table 2-1.

Table 2-1 Target system prerequisites

Program number	Product name and minimum VRM/Service level
5650 z/OS	► z/OS V2.1 with PTF for APAR OA48650 or higher ► z/OS V2.2 with PTF for APAR OA48650 or higher
5650 z/OS	► RACF V2.1 with PTF for APAR OA48359 or higher ► RACF V2.2 with PTF for APAR OA48359 or higher

RSA Authentication Manager 8.1 is required if the RSA plug-in is used.

If you want to install IBM MFA for z/OS into its own SMP/E environment, you can use the sample jobs that are provided to perform part or all of the installation tasks. The SMP/E jobs assume that all DDDEF entries that are required for SMP/E execution were defined in appropriate zones.

2.1.2 SMP/E considerations for installing IBM MFA for z/OS

Use the SMP/E **RECEIVE**, **APPLY**, and **ACCEPT** commands to install this release of IBM MFA for z/OS.

Note: We specifically chose to install the MFA product in your z/OS SMP/E CSI where the RACF security server is installed. This configuration ensures that all of the MFA corequisites and prerequisites with RACF also are installed with MFA during the SMP/E **Apply** and **Accept** steps.

The RIMLIB data set can be extracted as indicated in the program directory by running the GIMUNZIP utility, as shown in Example 2-1.

Example 2-1 Extracting the RIMLIB data set

```
//UNZIP    EXEC PGM=GIMUNZIP,REGION=0M,PARM='HASH=NO'
//SYSUT3   DD UNIT=SYSALLDA,SPACE=(CYL,(50,10))
//SYSUT4   DD UNIT=SYSALLDA,SPACE=(CYL,(25,5))
//*SMPJHOME DD PATH='/usr/lpp/java/J1.4/'
//*SMPCPATH DD PATH='/usr/lpp/smp/classes/'
//SMPOUT   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SMPDIR   DD PATH='/smpe/smpnts/STP41582',
//            PATHDISP=KEEP
//SYSIN    DD *
<GIMUNZIP>
<ARCHDEF
name="S0002.CSP.STP41582.DOCLIB.pax.Z"
volume="MOXTMP"
newname="PRICHAR.MFA.DOCLIB">
</ARCHDEF>
<ARCHDEF
name="S0003.CSP.STP41582.RIMLIB.pax.Z"
volume="MOXTMP"
newname="PRICHAR.MFA.RIMLIB">
</ARCHDEF>
<ARCHDEF
name="S0005.CSP.STP41582.PGMDIR.pax.Z"
volume="MOXTMP"
newname="PRICHAR.MFA.PGMDIR">
//SMPCNTL  DD  *
  SET      BOUNDARY (GLOBAL) .
  RECEIVE
   FROMNTS(STP41582)
  /*    DELETEPKG
      .
/*
```

When you run RIMLIB(UNZIPJCL) to extract the product's relfile from archives, you create a sample job data set that is named `prichar.mfa.sample.job`, as shown in Example 2-2.

Example 2-2 Creating the sample job data set

```
//*-------------------------------------------------------------
//UNZIP    EXEC PGM=GIMUNZIP,REGION=0M,PARM='HASH=NO'
//SYSUT3   DD UNIT=SYSALLDA,SPACE=(CYL,(10,10))
//SYSUT4   DD UNIT=SYSALLDA,SPACE=(CYL,(15,5))
//*SMPJHOME DD PATH='/usr/lpp/java/J1.4/'           <===NOTE 2
//*SMPCPATH DD PATH='/usr/lpp/smp/classes/'         <===NOTE 2
//SMPOUT   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SMPDIR   DD PATH='/smpe/smpnts/STP41582/SMPRELF',   <===NOTE 3
//            PATHDISP=KEEP
//SYSIN    DD *
<GIMUNZIP>
<ARCHDEF
name="CBCACHE.IBM.HMFA110.F2.pax.Z"
volume="MOXSMP"
newname="prichar.mfa.sample.jobs">
</ARCHDEF>
</GIMUNZIP>
/*
```

Sample jobs

The following jobs are listed in the program directory's Sample Jobs section:

- AZFJ3ALO: ALLOCATE: Allocates target and distribution libraries
- AZFJ1SMA: SMP/E: Allocates SMP/E data sets
- AZFJ2SMI: SMP/E: Allocates CSI
- AZFJ4DDF: DDDEF: Defines SMP/E DDDEFs 2
- AZFJ5REC: RECEIVE: A sample RECEIVE job
- AZFJ6APP: APPLY: A sample APPLY job
- AZFJ7ACC: ACCEPT: A sample ACCEPT job

When you order the MFA product, you receive a package that contains the FMID function HMFA110 and all PTFs that are available at the time of ordering. The following PTFs were shipped in our package:

- UI40279
- UI41424
- UI38845
- UI39463

Two of those PTFs (UI38845 and UI40279) introduce changes in the installation process of the MFA product. The changes that are described in the PTF UI38845 hold data are shown in Example 2-3 on page 17.

Example 2-3 PTF UI38845 installation changes

```
++ HOLD(UI38845) SYS FMID(HMFA110) REASON(ACTION) DATE(16174)
   COMMENT
    (************************************************************
     * FUNCTION AFFECTED: Multi-Factor Authentication (PI60774)  *
     ************************************************************
     * DESCRIPTION : ACTION                                      *
     ************************************************************
     * TIMING : Pre-APPLY                                        *
     ************************************************************
     Shut down your MFA started task.
     ************************************************************
     * FUNCTION AFFECTED: Multi-Factor Authentication (PI60774)  *
     ************************************************************
     * DESCRIPTION : ACTION                                      *
     ************************************************************
     * TIMING : Post-APPLY                                       *
     ************************************************************
     Customize and then run SAZFSAMP members AZFALLOC, AZFISMKD, and
     AZFDDEF to update your SMP/E environment to support the
     installation of subsequent PTFs.

     Allocate New SMP/E Target and Distribution Libraries Edit and
     submit sample job AZFALLOC to allocate the new SMP/E
     distribution library and optionally create a new zFS data set for
     IBM Multi-Factor Authentication for z/OS.  This job must be
      submitted by a user who has authority to create and format zFS
      aggregates.  Consult the instructions in the sample job for more
      information.

      Allocate File system Paths
      The target system zFS data set must be mounted on the driving
      system when running the sample AZFISMKD job since the job will
      create paths in the zFS.  To mount the file system issue the
      following command from OMVS while running as superuser (root):
      mount -t ZFS -f <aggregate_name> <mount_point>
      zFS must be active on the driving system if you are installing
      IBM Multi-Factor Authentication for z/OS into a file
      system that is zFS.

      The recommended mount point is /usr/lpp/azfv1r1. However, if this
      is not the mount point, a directory tree of the form
      -PathPrefix-/usr/lpp/azfv1r1 must exist or significant tailoring
      of the AZFISMKD, AZFMKDIR, and AZFDDEF files will be required.
      Edit and submit sample job AZFISMKD to create the file system
      for IBM Multi-Factor Authentication for z/OS.  Consult the
      instructions in the sample job for more information. Note that
      this job must be submitted by a user who has READ authority to
       BPX.SUPERUSER.  If you create a new file system for this
       product, consider updating the BPXPRMxx PARMLIB member to
       mount the new file system at IPL time. This action can be
       helpful if an IPL occurs before the installation is
       completed.
```

PTF UI40279 introduces another change in the installation path, which is being changed from /usr/lpp/azfv1r1 to /usr/lpp/IBM/azfv1r1, as shown in Example 2-4.

Example 2-4 PTF UI40279 changes to installation process

```
APAR Identifier ...... PI67593      Last Changed ........ 16/09/02
UPDATE SMP/E JOBS FOR REVISED INSTALLATION
PATH FOR MFA IHSA MODULES.

Symptom ...... NF NEWFUNCTION       Status ........... CLOSED  UR1
Severity ................... 3      Date Closed ......... 16/08/22
Component .......... 565516201      Duplicate of ........
Reported Release ......... 110      Fixed Release ............ 999
Component Name MULTI-FACTOR AU      Special Notice
Current Target Date ..16/11/13      Flags
Release 110   : UI40279 available 16/08/25 (F608 )

ERROR DESCRIPTION:
To conform with new packaging requirements
the installation path has been changed from
/usr/lpp/azfv1r1 to /usr/lpp/IBM/azfv1r1.
Additionally, the IBM directory subordinate
the modules directory has been changed to be
parallel to the modules directory.
.
AZFDDDEF and AZFMKDIR have been updated to use the
revised HFS directory structure for the MFA IHSA modules.
```

Three new installation jobs are now included with this PTF. Those jobs must be executed before the product is applied and accepted to avoid any installation error during the AZFALLOC, AZFISMKD, and AZFDDEF SMP/E steps.

When you run the receive job, you can see that the SAZFSAMP library member AZF$INDX contains the sample jobs, as shown in Example 2-5.

Example 2-5 New jobs included in the installation process

```
BROWSE    AZF.SAZFSAMP(AZF$INDX)                  Line 0000000019 C
 Command ===>                                                Scroll
Member   - Description
AZF#IN00 - JCL to invoke the AZF Authentication Started Task.
AZF#IN01 - JCL to invoke the AZF Registration Web Server.
AZFALLOC - JCL to allocate additional SMP/E libraries.
AZFDDDEF - JCL to update SMP/E environment with new libraries.
AZFISMKD - JCL to populate zFS directory structure.
AZFJ1SMA - JCL to allocate SMP/E data sets.
AZFJ2SMI - JCL to initialize the SMP/E CSI.
AZFJ3ALO - JCL to allocate the distribution and target libraries.
AZFJ4DDF - JCL to add the necessary DDDEF entries.
AZFJ5REC - JCL to perform the SMP/E RECEIVE function for the product.
AZFJ6APP - JCL to perform the SMP/E APPLY function for the product.
AZFJ7ACC - JCL to perform the SMP/E ACCEPT function for the product.
AZFMKDIR - EXEC to create directories.
```

2.1.3 Installing the program directory

Complete the following steps to install the program directory:

1. Allocate SMP/E target and distribution libraries. Edit and submit sample job AZFJ3ALO to allocate the SMP/E target and distribution libraries for IBM (step 6.1.6 in the program directory).
2. After running the AZFJ3ALO job, now also run the new AZFALLOC - JCL to allocate extra SMP/E libraries. This job allocates two new data sets, one zFS file system, and one distribution library.

   Note: Allocating a new HFS to contain the Apache modules is optional. If you do not want to create a file system and use directories in a file system, remove the allocation for SAZFAMOD. We chose to allocate these two new data sets.

   After running the jobs, that following data sets are allocated:

   – AZF.AAZFAMOD
   – AZF.AAZFEXEC
   – AZF.AAZFLOAD
   – AZF.AAZFMENU
   – AZF.AAZFPENU
   – AZF.AAZFSAMP
   – AZF.AAZFTENU
   – AZF.SAZFAMOD
   – AZF.SAZFAMOD.DATA
   – AZF.SAZFEXEC
   – AZF.SAZFLOAD
   – AZF.SAZFMENU
   – AZF.SAZFPENU
   – AZF.SAZFSAMP
   – AZF.SAZFTENU
3. You must now mount the new ZFS file system by using the **MOUNT** command. Update your BPXPRMxx PARMLIB member as shown in Example 2-6 and issue MVS command **SET OMVS=xx** to activate it.

   Example 2-6 Mounting the ZFS file system

```
/*AZF: MFA            */
  MOUNT FILESYSTEM('AZF.SAZFAMOD')
        MOUNTPOINT('/usr/lpp/IBM/azfv1r1/')
        TYPE(ZFS) MODE(RDWR)
```

4. Verify that the filesystem was mounted successfully by using the **D OMVS,F** command. The confirmation that the mount command successfully ran is shown in Example 2-7.

   Example 2-7 Confirmation of the mounted filesystem

```
BPXO045I 10.38.41 DISPLAY OMVS 159
OMVS     0010 ACTIVE             OMVS=(00)
TYPENAME   DEVICE ----------STATUS----------- MODE  MOUNTED    LATCHES
ZFS           103 ACTIVE                      RDWR  09/20/2016  L=148
  NAME=AZF.SAZFAMOD                                 16.07.24    Q=0
  PATH=/Z22RO1/usr/lpp/IBM/azfv1r1
  OWNER=X9       AUTOMOVE=Y CLIENT=N
```

5. You must now populate this new zFS filesystem by running a new job in AZF.SAZFSAMP(AZFISMKD), as shown in Example 2-8. This JCL creates directories in the HFS for IBM MFA for z/OS V1R1 by invoking AZFMKDIR. We used the usr/lpp/IBM/azfv1r default directory PATH mount point.

Example 2-8 Populating the new ZFS filesystem

```
//SYSEXEC DD DSN=AZF.SAZFSAMP,
//      DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
PROF MSGID
AZFMKDIR
/*
//
```

6. Verify that the directory structure that is shown in Example 2-9 was created.

Example 2-9 Expected directory structure

```
Select one or more files with / or action codes.
EUID=0   /Z22RO1/usr/lpp/IBM/azfv1r1/
  Type  Perm  Changed-GMT-1     Owner      ------Size  Filename    Row 1 of 4
_ Dir    755  2016-09-20 16:07  SAMBA            8192  .
_ Dir    755  2016-09-19 13:44  SAMBA            8192  ..
_ Dir    755  2016-09-20 16:07  SAMBA            8192  modules
_ Dir    755  2016-09-20 16:07  SAMBA            8192  IBM
```

7. The program directory now instructs you to create DDDEF Entries (6.1.7). Edit and submit sample job AZFJ4DDF to create DDDEF entries for the SMP/E target and distribution.
8. You must run new job AZF.SAZFSAMP(AZFDDDEF), which executes the SMP/E UCLIN to create the DDDEF entries for the new zFS and distribution libraries, as shown in Example 2-10.

Example 2-10 DDDEF entries for new zfs and distribution libraries

```
//DDDEFD   EXEC PGM=GIMSMP,REGION=OM
//SMPCSI   DD  DSN=MVSSMPE.GLOBAL.CSI,
//             DISP=SHR
//SMPCNTL  DD *
 SET BDY(MVSD100).
 UCLIN .
   ADD DDDEF(AAZFAMOD)
       DATASET(AZF.AAZFAMOD)
       UNIT(SYSALLDA)
   /*  VOLUME(#volserd)                    */
       WAITFORDSN
       SHR
       .
 ENDUCL.
//DDDEFT   EXEC PGM=GIMSMP,REGION=OM
//SMPCSI   DD  DSN=MVSSMPE.GLOBAL.CSI,
//             DISP=SHR
//SMPCNTL  DD *
 SET BDY(MVST100).
 UCLIN .
   ADD DDDEF(SAZFAMOD)
```

```
     PATH('/usr/lpp/IBM/azfv1r1/IBM/')
       .
 ENDUCL.
/*
```

9. If you are using the z/OS SMP/E environment where RACF was installed, you can skip jobs (AZFJ1SMA and AZFJ2SMI) that allocate the dedicated SMP/E data sets for MFA.
10. Run the following jobs to perform SMP/E **Receive, Apply**, and Accept functions:
    - AZFJ5REC
    - AZFJ6APP
11. During the APPLY CHECK and APPLY processes, add the GOUPEXTEND option and specify BYPASS(HOLDSYSTEM(ACTION,DOC,RESTART)), as shown in Example 2-11.

Example 2-11 Applying check and applying process parameters

```
//SMPCNTL   DD  *
  SET BDY(MVST100) .                          /* NOTE 2 */
  APPLY GROUPEXTEND
        SELECT(                              /* NOTE 3 */
               HMFA110
              )
        FORFMID(                             /* NOTE 3 */
               HMFA110
               )
    CHECK
   BYPASS(HOLDSYSTEM(ACTION,DOC,RESTART))
   COMPRESS(ALL)
     .
/*
```

12. During the ACCEPT CHECK and ACCEPT processes, add the GOUPEXTEND option and specify BYPASS(HOLDSYSTEM(ACTION,DOC,RESTART)), as shown in Example 2-12.

Example 2-12 Accepting check and accepting process parameters

```
//SMPCNTL   DD  *
   SET BDY(MVSD100) .                         /* NOTE 2 */
   ACCEPT GROUPEXTEND
          SELECT(                            /* NOTE 3 */
                 HMFA110
                )
          FORFMID(                           /* NOTE 3 */
                 HMFA110
                 )
    BYPASS(HOLDSYSTEM(ACTION,DOC,RESTART))
    COMPRESS(ALL)
      .
 /*
```

13. Ensure that the SMP/E report shows the product and its associated PTFs are applied and accepted, as shown in Example 2-13.

Example 2-13 Confirmation of the SMP/E installation process

```
HMFA110  ACCEPTED  FUNCTION  HMFA110

UI38845  ACCEPTED  PTF       HMFA110  HOLDS   *ACTION(UI38845)    *DOC(UI38845)

UI39463  ACCEPTED  PTF       HMFA110  PRE      UI38845
                                      HOLDS   *ACTION(UI39463)    *DOC(UI39463)

UI40279  ACCEPTED  PTF       HMFA110  PRE      UI38845
                                      HOLDS   *ACTION(UI40279)    *DOC(UI40279)
```

Although the base installation is now complete, MFA must be customized before it can be used. The customization process is described next.

2.2 Basic customization

After you install IBM MFA, you must customize the product for your environment. This customization must be completed before you run IBM MFA for the first time.

In this section, we describe the setup steps that we followed to implement the MFA base functions.

We do not intend to reproduce the information that you find in *IBM Multi-Factor Authentication for z/OS Installation and Customization*, SC27-8447-01. Rather, we want to complement the information of that publication and provide you with some examples, scripts, and screen shots of the implementation that we carried out on our system.

2.2.1 Preparation and customization overview

Ensure that you are using the latest version of the following publications, which correspond to MFA V1.1.0:

- *IBM Multi-Factor Authentication for z/OS Installation and Customization*, SC27-8447-01:
  `http://publibz.boulder.ibm.com/epubs/pdf/azfic101.pdf`
- *z/OS Cryptographic Services ICSF System Programmer's Guide*, SA22-7520-17

Chapter 1 of the *IBM Multi-Factor Authentication for z/OS Installation and Customization* publication provides an overview of the system programming steps that are required to implement MFA. The steps are divided into systems programming and RACF administration steps.

Systems programming

The following process is used to program the system:

1. Copy SAZFEXEC(AZFEXEC).
2. Customize AZFEXEC.
3. Copy SAZFSAMP(AZF#IN00) and SAZFSAMP(AZF#IN01).
4. Customize AZF#IN00 and AZF#IN01.

5. Authorize the Load Library.
6. Update SCHEDxx PARMLIB program properties.

RACF administration

The following process is used to administer the RACF:

1. Define a user for AZF started task.
2. Define the entry in STARTED class.
3. Activate MFADEF class.

2.2.2 Customization

Complete the following steps:

1. Copy `AZF.SAZFEXEC(AZFEXEC)` member to a data set in your SYSEXEC concatenation.
2. Customize the `azfhlq` parameter of the `AZFEXEC` member of the data set to reflect the HLQ we used for the MFA libraries (default is AZF), as shown in Example 2-14.

Example 2-14 Customizing the azfhlq parameter in AZFEXEC

```
/* rexx */
/*************************************
/*  5655-162
/*  © Rocket Software, Inc. or its affiliates
/*  All Rights Reserved.
/*************************************
/*
/* MODIFICATION INSTRUCTIONS:
/*
/* - CHANGE ?AZFHLQ? to the high-level qu
/*   IBM Multi-Factor Authentication for
/*
/*************************************

Arg PARMS

azfhlq   = 'AZF'
...
```

3. Copy `AZF.SAZFSAMP(AZF#IN00)` and `AZF.SAZFSAMP(AZF#IN01)` to the PROCLIB from which you run started tasks. You can use the **`$D PROCLIB`** command to identify for a PROCLIB data set where to copy the two members. Ensure that the **`Start`** command picks up your chosen PROCLIB. Consider the following points:
   - `AZF#IN00` is the IBM MFA started task
   - `AZF#IN01` is the started task for the registration server (“imbedded” Apache webserver)

   Choose one of the PROCLIB data sets in your installation PROCLIB concatenation.
4. Customize the PROCLIB members `AZF#IN00` and `AZF#IN01` for the high-level qualifier.

   Edit `AZF#IN00` and `AZF#IN01` and change `azfhlq` to the high-level qualifier of where you installed IBM MFA in the STEPLIB DD statement, as shown in Example 2-15.

Example 2-15 PROCLIB member with changed HLQ

```
//AZF#IN00 PROC
//*****************************************************************/
//* 5655-162                                                      */
```

```
//* © Rocket Software, Inc. or its affiliates 2015, 2016.           */
//* All Rights Reserved.                                            */
//*****************************************************************/
//* AZF0001 - Initial version                                       */
//*****************************************************************/
//* DESCRIPTION:                                                    */
//*   Started Task Sample JCL                                       */
//*****************************************************************/
//*                                                                 */
//* MODIFICATION INSTRUCTIONS:                                      */
//*                                                                 */
//* - CHANGE ?AZFHLQ? to the high-level qualifier of the            */
//*   IBM Multi-Factor Authentication for z/OS product libraries.   */
//*                                                                 */
//*****************************************************************/
//AZF110  EXEC PGM=AZFSTCMN,
//             REGION=128M,
//             TIME=1440
//STEPLIB  DD  DISP=SHR,DSN=AZF.SAZFLOAD         <==HLQ changed
//SYSPRINT DD  SYSOUT=*
```

5. Authorize the Load Library by updating your PROGxx PARMLIB member (see Example 2-16) and activate the update by using the MVS **SET PROG=xx** command. Replace the VOLUME() parameter with the actual VOLSER of the volume where you allocated the SAZFLOAD data set.

Example 2-16 PROGxx member to update APF

```
APF ADD DSNAME(AZF.SAZFLOAD)                              VOLUME(Z21SMP)
```

Note: If your AZFLOAD data set is SMS-managed, you can replace "VOLUME(xxxxxxx)" with "SMS".

Run the **D PROG,APF** command to verify that the library was successfully APF-authorized, as shown in Example 2-17.

Example 2-17 Confirming APF update

```
D PROG,APF
CSV450I 10.29.04 PROG,APF DISPLAY 010
FORMAT=DYNAMIC
ENTRY VOLUME DSNAME
   1  Z22R01 SYS1.LINKLIB
   2  Z22R01 SYS1.SVCLIB
   3  Z22R01 SYS1.CMDLIB
   4  MNR2R3 SYS1.COB2CICS
   5  Z22R01 SYS1.COB2LIB
...
157  Z21SMP AZF.SAZFLOAD
```

6. Update the SCHEDxx parmlib to identify the program properties for AZFSTCMN, which requires special attributes. The Program Properties Table (PPT) statement within the SCHEDxx member specifies a list of programs that require special attributes.
7. Edit your active PARMLIB SCHEDxx member that defines program properties (in your PARMLIB concatenation, as reported by using the **D PARMLIB** command).

Add the entry that is shown in Example 2-18.

Example 2-18 Updating the PPT

```
MT SIZE(64K)                            /* MASTER TRACE TBL SIZE      */
PPT PGMNAME(EDGMAIN)                   /* PROGRAM NAME               */
    NODSI                               /* PROTECTION KEY             */
 /*                                                                    */
 /* MFA                                                                */
PPT PGMNAME(AZFSTCMN) /* MULTI-FACTOR AUTH */
    KEY(2) /* PROTECTION KEY */
    NOSWAP /* NON-SWAPABLE */
    CANCEL /* CANCELABLE */
```

8. Save and activate the changes by issuing the **T SCH=xx** command, as shown in Example 2-19.

Example 2-19 Updating the PPT

```
T SCH=00
IEE252I MEMBER  SCHED00 FOUND IN SYS1.PARMLIB9
IEF729I MT STATEMENT IGNORED, NOT SUPPORTED FOR DYNAMIC UPDATE.
IEE536I SCH      VALUE 00 NOW IN EFFECT
```

9. Verify the PPT update by using the **D PPT** command to display the contents of the PPT. It should show the AZFSTCMN entry as shown in Example 2-20.

Example 2-20 Confirming PPT

```
D PPT
IEF386I 10.38.47 DISPLAY PPT 065
Parmlib Values
PgmName  NC NS PR ST ND BP Key 2P 1P NP NH CP
APSPPIEP  .  Y  .  Y  .  .  1   .  .  .  .  .
AZFSTCMN  .  Y  .  .  .  .  2   .  .  .  .  .
```

RACF administration

After you complete the initial system programming steps, complete the following steps:

1. Define a user for the AZF started task with the following properties:
   - No passphrase or password
   - Owned by a suitable started task group
   - PROTECTED
   - No TSO segment
   - An OMVS segment with a unique user ID

   The RACF **ADDGROUP** and **ADDUSER** commands that are used are shown in Example 2-21.

Example 2-21 Defining and configuring the AZF group and user started task

```
//STEP01   EXEC PGM=IKJEFT1A
//SYSTSPRT DD SYSOUT=A
//SYSTSIN  DD *
ADDGROUP AZFGRP SUPGROUP(SYS1) OWNER(SYS1) OMVS(GID(88887))
ADDUSER  AZFSTC  DFLTGRP(AZFGRP) NOPASSWORD -
         OMVS(UID(15100) PROGRAM(/bin/sh) -
         HOME(/usr/lpp/IBM/azfv1r1))
```

To verify the user information, use the RACF **LISTUSER (LU)** command. The response from the command is shown in Example 2-22.

Example 2-22 Verifying the started task information

```
LU AZFSTC OMVS
USER=AZFSTC  NAME=UNKNOWN  OWNER=PRICHAR   CREATED=16.263
  DEFAULT-GROUP=AZFGRP   PASSDATE=N/A    PASS-INTERVAL=N/A PHRASEDATE=N/A
  ATTRIBUTES=PROTECTED
  REVOKE DATE=NONE   RESUME DATE=NONE
LAST-ACCESS=16.278/09:00:26
  CLASS AUTHORIZATIONS=NONE
  NO-INSTALLATION-DATA
  NO-MODEL-NAME
  LOGON ALLOWED   (DAYS)          (TIME)
  -----------------------------------------
  ANYDAY                          ANYTIME
   GROUP=AZFGRP    AUTH=USE      CONNECT-OWNER=PRICHAR   CONNECT-DATE=16.263
     CONNECTS=    29  UACC=NONE     LAST-CONNECT=16.278/09:00:26
     CONNECT ATTRIBUTES=NONE
     REVOKE DATE=NONE   RESUME DATE=NONE
 SECURITY-LEVEL=NONE SPECIFIED
 CATEGORY-AUTHORIZATION
  NONE SPECIFIED
 SECURITY-LABEL=NONE SPECIFIED

 OMVS INFORMATION
 ----------------
 UID= 0000015100
 HOME= /usr/lpp/IBM/azfv1r1
 PROGRAM= /bin/sh
 CPUTIMEMAX= NONE
 ASSIZEMAX= NONE
 FILEPROCMAX= NONE
 PROCUSERMAX= NONE
 THREADSMAX= NONE
 MMAPAREAMAX= NONE
 ***
```

2. Define an entry in the RACF STARTED class to ensure that the IBM MFA address space features the proper level of authority by using the RACF **RDEFINE** command to define it and the **SETROPTS** command to activate the definition, as shown in Example 2-23.

Example 2-23 Started task definition

```
//STEP01   EXEC PGM=IKJEFT1A
//SYSTSPRT DD SYSOUT=A
//SYSTSIN  DD *
RDEFINE  STARTED AZF*.** -
         STDATA(USER(AZFSTC) GROUP(AZFGRP))
SETROPTS RACLIST(STARTED) REFRESH
```

3. Activate the MFADEF class if the class is not yet activated. The MFADEF class must be active before a user can log on with IBM MFA, as shown in Example 2-24.

Example 2-24 Activating the MFA class

```
//STEP01   EXEC PGM=IKJEFT1A
//SYSTSPRT DD SYSOUT=A
//SYSTSIN  DD *
/* activate the MFADEF Class:   */
SETROPTS CLASSACT(MFADEF) RACLIST(MFADEF) GENERIC(MFADEF)
//*
```

4. Verify that the class is active and RACLISTed by using the **SETROPTS** command. The response is shown in Example 2-25.

Example 2-25 Verifying MFADEF class is active

```
ATTRIBUTES = INITSTATS WHEN(PROGRAM -- BASIC) SAUDIT CMDVIOL NOOPERAUDIT
STATISTICS = DATASET APPL DASDVOL GDASDVOL GTERMINL GZMFAPLA PTKTDATA
             PTKTVAL RACFHC RACHCMBR TAPEVOL TERMINAL ZMFAPLA
ACTIVE CLASSES = DATASET USER GROUP ACCTNUM ACICSPCT AIMS APPCLU APPCPORT
                 APPCSERV APPL BCICSPCT CBIND CCICSCMD CDT CFIELD CIMS
                 CPSMOBJ CPSMXMP CRYPTOZ CSFKEYS CSFSERV DASDVOL DCICSDCT
                 DIGTCERT DIGTRING DIMS DSNADM DSNR ECICSDCT EJBROLE FACILITY
                 FCICSFCT FIELD GCICSTRN GCPSMOBJ GCSFKEYS GDASDVOL GEJBROLE
                 GIMS GLOBAL GMBR GSDSF GXCSFKEY GXFACILI GZMFAPLA HCICSFCT
                 IDIDMAP JCICSJCT KCICSJCT LOGSTRM MCICSPPT MFADEF NCICSPPT
                 NETCMDS OPERCMDS PCICSPSB PERFGRP PMBR PROGRAM PROPCNTL
                 PTKTDATA PTKTVAL QCICSPSB RACFHC RACHCMBR RCICSRES RDATALIB
                 SCICSTST SDSF SERVAUTH SERVER STARTED SURROGAT TAPEVOL
                 TCICSTRN TIMS TSOAUTH TSOPROC UCICSTST UNIXMAP VCICSCMD
                 VTAMAPPL WBEM WCICSRES XCSFKEY XFACILIT ZMFAPLA
...
SETR RACLIST CLASSES =  ACCTNUM APPCPORT APPCSERV APPL CBIND CDT CRYPTOZ
                        CSFKEYS CSFSERV DIGTCERT DIGTRING DSNR EJBROLE
                        FACILITY FIELD IDIDMAP MFADEF NETCMDS OPERCMDS
                        PERFGRP PROPCNTL PTKTDATA PTKTVAL RACFHC RDATALIB
                        SDSF SERVAUTH SERVER STARTED SURROGAT TSOAUTH
                        TSOPROC VTAMAPPL WBEM XCSFKEY XFACILIT
```

The basic customization is now complete.

3

Preparing for IBM TouchToken

In this chapter, we prepare for configuring the IBM TouchToken support. Several prerequisite tasks might be necessary, depending on your configuration's status.

This chapter includes the following topics:

3.1 Configuring MFA for IBM TouchToken

If you want configure IBM Multi-Factor Authentication (MFA) for IBM TouchToken, you must customize the following major components to the IBM TouchToken system:

- The AZFTOTP1 authentication load module that is loaded by the IBM MFA STC. The AZFTOTP1 load module must run in every instance of the IBM MFA STC where IBM TouchToken users log on.
- The IBM TouchToken registration web server that runs as a separate started task.

> **Note:** In a sysplex environment where the RACF database and ICSF TKDS are shared across member LPARs, the IBM TouchToken registration web server must run only on one LPAR in the sysplex.

- The IBM TouchToken iOS application that runs on TouchID-capable iOS devices.

The following configuration process was used:

1. Administer RACF for IBM TouchToken.
2. Prepare ICSF.
3. Configure a PKCS#11 token.
4. Configure an Application Transparent Transport Layer Security (AT-TLS) profile.
5. Test the TLS configuration.
6. Configure the IBM TouchToken.
7. Install the shared secret as a configuration profile on the Apple iOS device.

The following sections describe the configuration steps that we used.

3.2 RACF MFA basic configuration

Complete the following steps to define MFA factors and authorize access to profiles:

1. Define factors in the MFADEF class.

   Use the **RDEFINE** (**RDEF**) command to define the authentication factors in the MFADEF class. You define MFA factors by creating an MFADEF class profile that is named FACTOR.AZFTOTP1. Complete the following steps:

   a. Define the factors in the MFADEF class by issuing the following command:

   ```
   RDEF MFADEF FACTOR.AZFTOTP1
   ```

   b. Verify the change by using the **RLIST** command, as shown in the following example:

   ```
   RLIST MFADEF FACTOR.AZFTOTP1 MFA
   ```

2. Define factors in FACILITY class.

   Use the **RDEF** command to define the authentication factors in the class. You define MFA factors by creating FACILITY class profiles that are named `IRR.RFACTOR.MFADEF.AZFTOTP1` and `IRR.RFACTOR.USER`. Complete the following steps:

   a. Define the factors in the FACILITY class for AZFTOTP1 by issuing the following command:

   ```
   RDEF FACILITY IRR.RFACTOR.MFADEF.AZFTOTP1
   ```

   b. Define the factors in the FACILITY class for USER, as shown in the following example:

   ```
   RDEF FACILITY IRR.RFACTOR.USER UACC(NONE)
   ```

c. Verify the changes, as shown in the following example:

```
RLIST FACILITY IRR.RFACTOR.MFADEF.AZFTOTP1

RLIST FACILITY IRR.RFACTOR.USER
```

3. Authorize access to the `IRR.RFACTOR.MFADEF.AZFTOTP1` profile for the administrators who run the panels to define the AZFTOTP1 factor by issuing the following command:

```
PERMIT IRR.RFACTOR.MFADEF.AZFTOTP1 CLASS(FACILITY) ID(SYS1) ACCESS(ALTER)
```

4. Authorize access to the `IRR.RFACTOR.USER` profile by using the following command:

```
PERMIT IRR.RFACTOR.MFADEF.AZFTOTP1 CLASS(FACILITY) ID(AZFSTC) ACCESS(READ)
```

5. Authorize the user ID of the registration server started task `AZF#IN00` to the `IRR.RFACTOR.USER` profile, as shown in the following example:

```
PERMIT IRR.RFACTOR.USER ACCESS(UPDATE) CLASS(FACILITY) ID(AZFSTC)

SETROPTS RACLIST(FACILITY) REFRESH
```

6. Allow `UPDATE` access for the user ID of the registration server started task, as shown in the following example:

```
PERMIT IRR.RFACTOR.USER ACCESS(UPDATE) CLASS(FACILITY) ID(AZFSTC)
```

You can run the commands in a batch job if you prefer, as shown in Example 3-1.

Example 3-1 Batch job to define the controls for AZFTOTP factor

```
//STEP01   EXEC PGM=IKJEFT1A
//SYSTSPRT DD SYSOUT=A
//SYSTSIN  DD *
 /* Define the factors in the MFADEF class: */
 RDEF MFADEF FACTOR.AZFTOTP1
 /* . Verify the change. For example:     */
 RLIST MFADEF FACTOR.AZFTOTP1 MFA
 /* Define the factors in the FACILITY class */
 /*                                                              */
 /*Use RDEFINE to define the authentication factors in the class. */
 /*define MFA                                                    */
 /*factors by creating FACILITY class profiles named             */
 /*IRR.RFACTOR.MFADEF.AZFTOTP1                                   */
 /*and IRR.RFACTOR.USER.                                         */
 /*Procedure                                                     */
 /*1. Define the factors in the FACILITY class for AZFTOTP1:     */
   RDEF FACILITY IRR.RFACTOR.MFADEF.AZFTOTP1
 /*2. Define the factors in the FACILITY class for USER:         */
   RDEF FACILITY IRR.RFACTOR.USER UACC(NONE)
 /*Authorize access to IRR.RFACTOR.MFADEF.AZFTOTP1 profile       */
 /*Authorize the administrators who execute the panels to the    */
 /*IRR.RFACTOR.MFADEF.AZFTOTP1 profile.                          */
 PERMIT IRR.RFACTOR.MFADEF.AZFTOTP1 CLASS(FACILITY) ID(SYS1) +
   ACCESS(ALTER)

 PERMIT IRR.RFACTOR.MFADEF.AZFTOTP1 CLASS(FACILITY) ID(AZFSTC) +
     ACCESS(READ)

 /*Authorize the user ID of the registration server started task to/
 /*IRR.RFACTOR.USER profile.                                     */
  PERMIT IRR.RFACTOR.USER ACCESS(UPDATE) CLASS(FACILITY) ID(AZFSTC)
 SETROPTS RACLIST(FACILITY) REFRESH
```

```
 /*3. Verify the changes. For example:                              */
   RLIST FACILITY IRR.RFACTOR.MFADEF.AZFTOTP1
   RLIST FACILITY IRR.RFACTOR.USER
//*
```

3.3 ICSF tasks

After you complete the RACF administration tasks, you must perform other system programming tasks for IBM TouchToken. We start with Integrated Cryptographic Service Facility (ICSF).

ICSF must be installed, configured, and the started task started, as described in *z/OS Cryptographic Services ICSF System Programmer's Guide*, SA22-7520.

For more information about the ICSF support for PKCS #11 setting up the TKDS, see *z/OS Cryptographic Services ICSF Writing PKCS #11 Applications*, SC14-7510.

3.3.1 Preparing ICSF

ICSF is software that is part of the z/OS Base element is named z/OS Cryptographic Services. It works with the cryptographic hardware on the mainframe. Programmers can call ICSF services to perform cryptographic operations.

To use ICSF, you must ensure that the appropriate hardware is in place. The hardware on zEntreprise servers (z196 z12 IBM z13®) consists of a feature that is named CPACF (CP assist for cryptographic processors) and one or more cryptographic cards.

ICSF supports Cryptographic Coprocessors and the Crypto Express cards. These cards are secure, tamper-proof enclosures. ICSF provides access to the cards' functions.

For more information about and installing the latest Web Deliverable for your z/OS release, see the following website:

`http://www.ibm.com/servers/eserver/zseries/zos/downloads/`

If you do download a new version, follow the instructions that are provided to install the code before you proceed. Ensure that ICSF is at HCR77A0 or higher.

After the ICSF code is installed and available on your system, you must set up some z/OS parameters, RACF, started task procedure for ICSF, parameters, and VSAM clusters.

3.3.2 ICSF systems programming tasks

The following parmlib members must be updated:

- LNKLSTxx

  Add `SYS1.SCSFMOD0` to the MVS linklist. If you are using PROGxx for linklist, it resembles Example 3-2.

  Example 3-2 Updating the LNKLSTxx

```
LNKLST DEFINE NAME(LNKLSTxx)
LNKLST ADD NAME(LNKLSTxx) DSNAME(SYS1.SCSFMOD0)
```

Depending on how you configure your PARMLIB members, you might add it to more than one member. If you added a LNKLSTxx, it can be activated by using the following command:

```
LNKLST ACTIVATE NAME(LNKLSTxx)
```

- APF authorization

  The library must be APF-authorized. Therefore, you must add the library to PROGxx, as shown in Example 3-3.

Example 3-3 Adding PF authorization

```
APF ADD DSNAME(SYS1.SCSFMOD0) VOLUME(******)
```

- Authorized TSO commands

  ICSF features several authorized TSO commands, which requires an update to `SYS1.PARMLIB(IKJTSO00)`. You must add **`CSFDAUTH`** and **`CSFDPKDS`** commands to each of the `AUTHCMD` and `AUTHTSF` tables in IKJTSO00, as shown in Example 3-4.

Example 3-4 Authorizing TSO command

```
AUTHPGM NAMES(              /****** AUTHORIZED PROGRAMS ***********/ +
                            /*                                     */ +
   EIRGPTKT                 /* pass ticket generator              */ +
   CSFDAUTH                 /*  ICSF                               */ +
   CSFDPKDS                 /*  ICSF                               */ +
   CSYMINOR                 /*  OPC/A DIALOG SERVICE ROUTINE       */ +
   EDGHSKP                  /*  Rmm                                */ +
   EDGUTIL                  /*  Rmm                                */ +
```

3.3.3 ICSF RACF tasks

Determine what you want to name the ICSF started task. We assume that you name it ICSF. You need a started task User ID that you set up by using the RACF commands. The **`ADDUSER`** command defines a protected User ID for ICSF because it is to be used by the started task only and makes the profile owned by group SYS1. It gives the user the default group of SYS1. Your installation might require you to use a different DFLTGRP and OWNER.

ICSF includes the following RACF tasks:

- Create a started task User ID.

  The **`RDEFINE STARTED`** command tells RACF what User ID and group to assign to the started task ICSF when the **`START ICSF`** command is issued.

  The **`SETROPTS`** command refreshes the profiles for STARTED class, which is normally RACLISTed (that is, profiles that are cached in a data space).

  The following commands are used:

  - `ADDUSER ICSF NOPASSWORD NAME('ICSF Task') DFLTGRP(SYS1) OW(SYS1)`
  - `RDEFINE STARTED ICSF.** STDATA(USER(ICSF) GROUP(SYS1))`
  - `SETROPTS RACLIST(STARTED) REFRESH`

- Create a started task in SYS1.PROCLIB.

  You must set up a member that is named ICSF (or whatever you are naming your ICSF task) in SYS1.PROCLIB on the system where you want to start ICSF. IBM provides a sample procedure to which you copy and rename to ICSF. The provided member is in SYS1.SAMPLIB(CSF). The data set must be RECFM=FB, LRECL=80. A sample of the procedure is shown in Example 3-5.

Example 3-5 ICSF procedure

```
//*  Licensed Materials - Property of IBM
//*  5694-A01 Copyright IBM Corp. 2009
//CSF PROC
//CSF EXEC PGM=CSFINIT,REGION=0M,TIME=1440,MEMLIMIT=NOLIMIT
//CSFPARM DD DSN=SYS2.PARMLIB(CSFPRM00),DISP=SHR
```

3.3.4 ICSF parameters

The parameters for ICSF are in the data set that is specified in the CSFPARM DD statement. The default PARMLIB member name is CSFPRM00 and sample members are inclucded in SYS1.SAMPLIB that are named CSFPRM00, as shown in Example 3-6.

Example 3-6 Sample CSFPRM00 member

```
/*******************************************************************/
/*     LICENSED MATERIALS - PROPERTY OF IBM                        */
/*                                                                 */
/*     5650-ZOS                                                    */
/*                                                                 */
/*     COPYRIGHT IBM CORP. 1990, 2013                              */
/*                                                                 */
/*  THIS IS A SAMPLE OF THE ICSF OPTIONS DATASET                   */
/*                                                                 */
/*******************************************************************/
CKDSN(SYS2.CSF.CSFCKDS)
PKDSN(SYS2.CSF.CSFPKDS)
TKDSN(SYS2.CSF.CSFTKDS)
PKDSCACHE(64)
COMPAT(NO)
KEYAUTH(YES)
CHECKAUTH(NO)
TRACEENTRY(10000)
USERPARM(USERPARM)
REASONCODES(ICSF)
SSM(YES)
DOMAIN(0)
CTRACE(CTICSF00)
```

The following ICSF parameters are available:

- CKDSN: The VSAM key-sequenced data set that the cryptographic key data set (CKDS) where the symmetric cryptography data encryption standard (DES) and Triple-DES keys are stored.
- PKDSN: The VSAM key-sequenced data set that is named the public-key architecture (PKA) key data set (PKDS) where your private keys are stored. The PKA is the type of keys that are used in certificates and other asymmetric algorithms.

- TKDS: The TKDS is the repository for the keys that are used by PKCS#11. The TKDS is a key-sequenced VSAM data set. To enable applications to create and use persistent PKCS #11 tokens and objects that are using the PKCS #11 services, the TKDS must be allocated and the TKDS data set name must be specified on the TKDSN parameter of the options data set when you first start ICSF.
- DOMAIN(xx): If you assigned only one usage domain to the logical partition (LPAR) that is running ICSF, you can omit this parameter and the domain is detected automatically. If the LPAR includes multiple usage domains, you must select one domain and code the value in this parameter.
- SSM (special secure mode): This parameter must be set to YES if you want to use the key generation utility program (KGUP) to generate keys or import keys. Otherwise, you can set this parameter to NO.
- COMPAT(NO): This parameter is coded if at all possible.
- COMPAT(YES): This parameter indicates that you want ICSF to provide the API for the ancient Programmed Cryptographic Facility software from IBM. This software allows you to use old applications that use PCF without converting them to call ICSF. Although this configuration is tempting to use, it impairs your ability to make dynamic master key changes in ICSF as you must shut down ICSF. COMPAT(NO) allows seamless master key changes without disruption to the application.

3.3.5 CKDS PKDS and TKDS data sets

ICSF uses two data sets for permanent storage of keys. The CKDS is used to store DES keys in records that can be referenced later by label by programs that are using ICSF services.

To define the CKDS, use the JCL sample found in `SYS1.SAMPLIB(CSFCKDS)`. You define the data set and leave it empty. When master keys are entered, the data set is initialized, as shown in Example 3-7.

Example 3-7 Sample JCL to define a CKDS VSAM data set

```
//DEFINE  EXEC PGM=IDCAMS,REGION=4M
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
  DEFINE CLUSTER (NAME(SYS2.CSF.CSFCKDS)          -
                  VOLUMES(MVJ005)            -
                  RECORDS(100 50)            -
                  RECORDSIZE(252,252)        -
                  KEYS(72 0)                 -
                  FREESPACE(10,10)           -
                  SHAREOPTIONS(2,3))         -
            DATA (NAME(CSF.CSFCKDS.DATA)     -
                  BUFFERSPACE(100000)        -
                  ERASE                      -
                  WRITECHECK)                -
           INDEX (NAME(SYS2.CSF.CSFCKDS.INDEX))
/*
```

The PKDS is used to store public and private Rivest-Shamir-Adleman (RSA) keys that are used for PKA processing. An example of the use of PKA is digital certificates, which can be generated by using the RACF **`RACDCERT`** commands with the ICSF option so that the private key of the certificate is stored in the ICSF PKDS that is encrypted under the RSA master key.

To define the PKDS, use the JCL sample that is found in SYS1.SAMPLIB(CSFPKDS), as shown in Example 3-8.

Example 3-8 Sample JCL to define a PKDS VSAM data set

```
//DEFINE  EXEC PGM=IDCAMS,REGION=64M
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
  DEFINE CLUSTER (NAME(SYS2.CSF.CSFPKDS)          -
                  VOLUMES(MVJ005)            -
                  RECORDS(100 50)            -
                  RECORDSIZE(800,3800)       -
                  KEYS(72 0)                 -
                  FREESPACE(0,0)             -
                  SHAREOPTIONS(2,3))         -
            DATA (NAME(SYS2.CSF.CSFPKDS.DATA)     -
                  BUFFERSPACE(100000)        -
                  ERASE                      -
                  CISZ(8192)                 -
                  WRITECHECK)                -
           INDEX (NAME(SYS2.CSF.CSFPKDS.INDEX))
/*
```

After the CKDS and PKDS data sets are defined, complete the following tasks:

- Ensure that the ICSF started task User ID has UPDATE access to the CKDS and PKDS data sets.
- Take the data set names that are used for the clusters and enter them in the CKDSN and PKDSN parameters in SYS1.PARMLIB(CSFPRM00) as described in 3.3.4, “ICSF parameters” on page 34.

The TKDS is the repository for the keys that are used by PKCS#11 and must be a VSAM key-sequenced data set with variable length records. Allocate the TKDS on a permanently resident volume.

To enable applications to create and use persistent PKCS #11 tokens and objects that use the PKCS #11 services, the TKDS must be allocated and the TKDS data set name must be specified on the TKDSN parameter of the options data set when you first start ICSF.

To define the PKDS, use the JCL sample that is found in SYS1.SAMPLIB(CSFTKDS), as shown in Example 3-9.

Example 3-9 Sample JCL to define a TKDS VSAM data set

```
//DEFINE  EXEC PGM=IDCAMS,REGION=4M
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
  DEFINE CLUSTER (NAME(SYS2.CSF.CSFTKDS)          -
                  VOLUMES(MVJ005)            -
                  RECORDS(100 50)            -
                  RECORDSIZE(2200,32756)     -
                  KEYS(72 0)                 -
                  FREESPACE(0,0)             -
                  CONTROLINTERVALSIZE(32768) -
                  SHAREOPTIONS(2,3))         -
            DATA (NAME(SYS2.CSF.CSFTKDS.DATA)     -
                  BUFFERSPACE(100000)        -
```

```
                ERASE                       -
                WRITECHECK)                 -
          INDEX (NAME(SYS2.CSF.CSFTKDS.INDEX))
/*
```

The ICSF installation options data set contains the following options that are related to the token data set (TKDS):

- `TKDSN(datasetname)`: Identifies the VSAM data set that contains the TKDS. It must be specified for ICSF to provide PKCS#11 services.
- `SYSPLEXTKDS(YES|NO,FAIL(YES|NO)`: Specifies whether the TKDS includes sysplex-wide data consistency.

3.3.6 CSFSERV and CSFKEYS classes in RACF

To use SAF to control access to keys, services, and utilities, you create and maintain general resource profiles in the following classes:

- The CSFSERV class controls access to CCA and PKCS #11 services and ICSF TSO panel utilities.
- The CSFKEYS class controls access to CCA cryptographic keys. You create profiles in this class (based on the label by which the key is defined in the CKDS or PKDS) to set access authority for the keys.
- The XCSFKEY class controls authorization checks when the symmetric key export services are called.
- The CRYPTOZ class controls access to and defines policy for cryptographic information within PKCS #11 tokens. PKCS #11 tokens are used exclusively by ICSF's PKCS #11 callable services. These tokens are abstract containers that hold keys, certificates, and other related cryptographic information. PKCS #11 tokens often are explicitly created and assigned to specific applications. The profiles in the CRYPTOZ class determine this assignment and indicate what operations are permitted for a specific PKCS #11 token.

Note: You should also investigate activating the CSFSERV and CSFKEYS classes in RACF as these classes are required if you want any control over what services can be used and what keys different users can access. In most cases, you do not want a vital ATM key to be usable by anyone on the system.

Keys within the TKDS are not encrypted. Therefore, it is important that the security administrator create a RACF profile to protect the TKDS from unauthorized access.

ICSF uses profiles in the SAF CRYPTOZ class to control access to PKCS#11 tokens. The user ID of the HTTP server started task must have the following SAF access level for the defined PKCS#11 token:

- SO.token_name CONTROL
- USER.token_name UPDATE

The job that is shown in Example 3-10 on page 38 is an "authorize all" sample, which is intended only for illustration purposes.

Note: The sample job in Example 3-10 should not be used in a production environment because it does not provide the level of security and protection that often is required in such environments. You might use it as a base to refine to your environment's needs.

Example 3-10 Sample authorization job

```
//SYSPROGX JOB 30000000,'MVS  JOB CARD  ',MSGLEVEL=(1,1),
//    CLASS=A,MSGCLASS=Q,NOTIFY=&SYSUID,TIME=1440,REGION=0M
//ALTUSER  EXEC PGM=IKJEFT01,DYNAMNBR=30
//SYSLBC  DD DSN=SYS1.BRODCAST,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD  *
SETROPTS RACLIST(CSFKEYS)
SETROPTS RACLIST(CSFSERV)
SETROPTS RACLIST(XCSFKEY)
SETROPTS CLASSACT(CSFSERV)
SETROPTS CLASSACT(CSFKEYS)
SETROPTS CLASSACT(CRYPTOZ) GENERIC(CRYPTOZ) RACLIST(CRYPTOZ)
RDELETE  CRYPTOZ  *
  RDELETE  CSFSERV  *
  RDELETE  CSFKEYS  *
RDEFINE CRYPTOZ SO.INSES54.* UACC(NONE)
RDEFINE CRYPTOZ USER.INSES54.* UACC(NONE)
PERMIT SO.INSES54.* CLASS(CRYPTOZ) ID(INSES54) ACC(UPDATE)
PERMIT USER.INSES54.* CLASS(CRYPTOZ) ID(INSES54) ACC(CONTROL)
RDEFINE  CRYPTOZ  *  UACC(ALTER)
  RDEFINE  CSFSERV  *  UACC(ALTER)
  RDEFINE  CSFKEYS  *  UACC(ALTER)
SETROPTS RACLIST(CRYPTOZ) REFRESH
  SETROPTS RACLIST(CSFSERV) REFRESH
  SETROPTS RACLIST(CSFKEYS) REFRESH
/*
//
```

Comparing Secure, Clear, and Protected keys

The following key types are available:

- Secure keys

  These keys provide high security because the key material is protected by the master key. Master keys are loaded within the cryptographic coprocessor and are used to wrap and unwrap secure key material within the secure boundaries of the HSM. This configuration prevents secure key material from ever appearing in the clear.

- Clear keys

  When symmetric encryption is performed (TDES and AES) with clear keys, ICSF uses the CPACF to provide high performance. *Clear key* refers to key material that is in the clear, which means the clear key value appears within application storage and within the keystore.

- Protected keys

  These keys blend the security of the Crypto Express3/4/5 and the performance characteristics of the CPACF. Although a secure key is encrypted under a master key, a protected key is encrypted under a wrapping key that is uniquely created for each LPAR.

  Consider the following points:

  - The wrapping key is created whenever an LPAR is activated or reset. The following variations of the wrapping key are available:
    - DES/TDES operational keys
    - AES keys

- The wrapping key is stored in the hardware system area (HSA). The wrapping key is accessible only by using firmware and cannot be read by the operating system or applications (even if running authorized).
- If a CEXxx coprocessor is available, a protected key can begin life as a secure key. That is, ICSF retrieves the secure key from the CKDS and the clear value is recovered (decrypted from under the master key) and then that clear key is reencrypted (or wrapped) under the appropriate wrapping key for that LPAR. The rewrapping is managed by IBM z Systems® firmware with the CEXxx coprocessor (CEX#C). The clear key value is never visible in operating system or application storage.
- The use of a protected key is driven by the application when it invokes a CPACF API and specifies the label of a secure key. The secure key is retrieved from the CKDS and brought into the CEXxx coprocessor where it is decrypted from under the master key and reencrypted under the appropriate wrapping key. That wrapped key is passed back to ICSF, which uses it within the CPACF to perform encryption or decryption operations. The underlying clear value of the key is never in any address space; instead, it is inside only the secure hardware or the CPACF.
- A protected key can be a DES, TDES, or AES key. If the same underlying key value is used as a secure key, clear key, or protected key, the resulting ciphertext is identical. That is, the encryption operation generates the same results no matter the type of key that is used. The difference is the level of protection that is provided by the hardware or operating system for the key values.
- Protected keys are a hybrid solution, which provides most of a “best of both worlds” scenario because of the following characteristics:
  - Uses a combination of CPACF and CEX (that uses ICSF).
  - Stored keys that are in z/OS are still encrypted.
  - CEX call decrypts the key and reencrypts with “wrapping key”.
  - Copies a wrapping key to protected HSA memory.
  - Wrapped key is returned and used on CPACF calls.

3.3.7 Activating the ICSF ISPF panel dialog

Ensure that the following ICSF ISPF data sets are allocated on your ISPF session:

- SYSEXEC CSF.SCSFCLIO
- ISPLLIB CSF.SCSFMODO
- ISPMLIB CSF.SCSFMSGO
- ISPPLIB CSF.SCSFPNLO
- ISPSLIB CSF.SCSFSKLO
- ISPTLIB CSF.SCSFTLIB

Add an option (for example, IC) to your ISPF primary panel, as shown in Example 3-11.

Example 3-11 Sample ISPF primary menu with IC added

```
CUSTOMPAC MASTER APPLICATION MENU
OPTION ===>                                          SCROLL ===> CSR

                                                  USERID - IBMUSER
                                                  TIME   - 13:44

  IS  ISMF     - Interactive Storage Management Facility
  P   PDF      - ISPF/Program Development Facility
  IP  IPCS     - Interactive Problem Control Facility
  AZ  MFA      - IBM Multi-Factor Authentication for z/OS
```

```
  DI  DITTO    - Data Interfile Transfer, Testing and Operations
  SD  SDSF     - System Display and Search Facility
  R   RACF     - Resource Access Control Facility
  HC  HCD      - Hardware Configuration Definition
  BMB BMR BLD  - BookManager Build (Create Online Documentation)
  BMR BMR READ - BookManager Read (Read Online Documentation)
  BMI BMR INDX - BookManager Read (Create Bookshelf Index)
  SM  SMP/E    - SMP/E Dialogs
  IC  ICSF     - Integrated Cryptographic Service Facility
  OS  SUPPORT  - OS/390 ISPF System Support Options
  OU  USER     - OS/390 ISPF User Options
  RR  RRS      - RRS Resource Recovery Services application
  S   SORT     - DF/SORT Dialogs
  US  USS      - OS/390 UNIX System Services options
  X   EXIT     - Terminate ISPF using list/log defaults
```

The menu option IC should start ICSF, as shown in Example 3-12.

Example 3-12 Starting the ICSF application

```
IC,'PANEL(CSF@PRIM) NEWAPPL(ICSF)'
```

Alternatively, instead of starting a panel option, you can create a sample REXX EXEC that dynamically allocates all of the ICSF ISPF libraries and starts the ICSF dialog, as shown in Example 3-13. In the example, the name of the sample REXX EXEC is `CSFPRIM`.

Example 3-13 REXX EXEC option to start ICSF option

```
/*REXX---------------------------------------------------------------*/
/* ICSFINTERACTIVE                                                   */
/*-------------------------------------------------------------------*/

   "ALTLIB ACT APPL(CLIST) DS('CSF.SCSFCLIO')"
ADDRESS ISPEXEC
"LIBDEF ISPPLIB DATASET ID('CSF.SCSFPNLO')"
"LIBDEF ISPMLIB DATASET ID('CSF.SCSFMSGO')"
"LIBDEF ISPSLIB DATASET ID('CSF.SCSFSKLO')"
"LIBDEF ISPTLIB DATASET ID('CSF.SCSFTLIB')"
"LIBDEF ISPLLIB DATASET ID('CSF.SCSFMODO'",
                        "'CSF.SCSFMOD1')"

"SELECT PANEL(CSF@PRIM) NEWAPPL(ICSF) PASSLIB"

"LIBDEF ISPPLIB"
"LIBDEF ISPMLIB"
"LIBDEF ISPSLIB"
"LIBDEF ISPTLIB"
"LIBDEF ISPLLIB"

ADDRESS TSO
"ALTLIB DEACT APPL(CLIST)"
EXIT
```

3.3.8 Starting ICSF

After you have everything in place, you can start the ICSF started task by issuing the MVS command **START ICSF** (or whatever you called your started task procedure). Error messages are initialized and issued that indicate there are incorrect master keys because you did not yet enter any master key.

At this point, you cannot do much with ICSF because no master keys are entered. The accelerator and coprocessor crypto express cards also cannot do much. However, you might still use the CPAC and PKCS#11 services.

z/OS PKCS #11 supports two different keying models. The key that is stored in the TKDS can be clear or secure (protected by master key). z/OS PKCS #11 tokens can contain clear keys, secure keys, or a mixture of both.

Secure keys require a secure coprocessor and can use accelerator crypto express or coprocessor (CEXnC), but not CPACF because it does not support asymmetric keys.

Also, a clear key does not need PKCS11 coprocessor. If PKCS11 coprocessor is available (EP11 mode), a secure key is generated.

Secure keys require a secure coprocessor. The Crypto Express4/5 is an optional hardware feature that is available on IBM zEnterprise® EC12 / z13. The Crypto Express4/5 can be configured as a cryptographic accelerator, a secure CCA cryptographic coprocessor, or a secure PKCS #11 cryptographic coprocessor (which is also known as an Enterprise PKCS #11 coprocessor or EP11).

An Enterprise PKCS #11 coprocessor with an active master key must be available to generate and use secure PKCS #11 keys. Clear keys do not require a coprocessor.

The start of our ICSF started task is shown in Example 3-14.

Example 3-14 ICSF start

```
S ICSF
IRR813I NO PROFILE WAS FOUND IN THE STARTED CLASS FOR 338
        ICSF WITH JOBNAME ICSF. RACF WILL USE ICHRIN03.
$HASP100 ICSF     ON STCINRDR
IEF695I START ICSF     WITH JOBNAME ICSF     IS ASSIGNED TO USER IBMUSER
 , GROUP SYS1
$HASP373 ICSF     STARTED
IEF403I ICSF - STARTED - TIME=13.59.14
CSFO0230  CKDSN(CSF.X9.SCSFCKDS)
CSFO0230  PKDSN(CSF.X9.SCSFPKDS)
CSFO0230  TKDSN(CSF.X9.SCSFTKDS)
CSFO0230  PKDSCACHE(64)
CSFO0212 PKDSCACHE KEYWORD NO LONGER SUPPORTED.
CSFO0230  COMPAT(NO)
CSFO0230  KEYAUTH(YES)
CSFO0212 KEYAUTH KEYWORD NO LONGER SUPPORTED.
CSFO0230  CHECKAUTH(YES)
CSFO0230  TRACEENTRY(1000)
CSFO0212 TRACEENTRY KEYWORD NO LONGER SUPPORTED.
CSFO0230  USERPARM(USERPARM)
CSFO0230  COMPENC(DES)
CSFO0212 COMPENC KEYWORD NO LONGER SUPPORTED.
CSFO0230  REASONCODES(ICSF)
```

```
CSFO0230  SSM(YES)
CSFO0230  DOMAIN(0)
IEE538I CTICSFOO MEMBER NOT FOUND IN PARMLIB
CSFO0424 CTRACE PARMLIB MEMBER CTICSFOO COULD NOT BE USED. SWITCHING TO
DEFAULT SETTINGS.
CSFM607I A CKDS KEY STORE POLICY IS NOT DEFINED.
CSFM607I A PKDS KEY STORE POLICY IS NOT DEFINED.
CSFM610I GRANULAR KEYLABEL ACCESS CONTROL IS DISABLED.
CSFM611I XCSFKEY EXPORT CONTROL FOR AES IS DISABLED.
CSFM611I XCSFKEY EXPORT CONTROL FOR DES IS DISABLED.
CSFM612I PKA KEY EXTENSIONS CONTROL IS DISABLED.
CSFM654I KEY ARCHIVING USE CONTROL IS DISABLED.
CSFM653I TKDS LOADED 29 RECORDS WITH AVERAGE SIZE 1688
CSFM015I FIPS 140 SELF CHECKS FOR PKCS11 SERVICES SUCCESSFUL.
CSFM506I CRYPTOGRAPHY - THERE IS NO ACCESS TO ANY CRYPTOGRAPHIC
COPROCESSORS OR ACCELERATORS.
CSFM126I CRYPTOGRAPHY - FULL CPU-BASED SERVICES ARE AVAILABLE.
CSFM001I ICSF INITIALIZATION COMPLETE
CSFM640I ICSF RELEASE FMID=HCR77B0.
```

The messages CSFM653I and CSFM015I that are shown in Example 3-14 on page 41 indicate that PKCS#11 services are now available.

A further description of Master keys, how to enter them, and how to start the key data sets, are beyond the scope of this publication because these topics are not relevant to our MFA quick setup. For more information about key entry methods, see *z/OS Cryptographic Services ICSF Administration Guide*, SC14-7506.

3.3.9 Validating PKCS#11 services

You can use the utility program testpkcs11 for troubleshooting purposes. IBM provides a sample PKCS #11 program that is named testpkcs11. The program is passed the name of a PKCS #11 token and performs the following steps:

1. Creates a token that includes the passed name.
2. Generates an RSA key-pair.
3. Encrypts some test data by using the public part of the key-pair.
4. Decrypts the data by using the private part of the key-pair.
5. Deletes the key-pair and the token.

You can use this program as a utility program to test the system configuration for PKCS #11 and troubleshoot problems. IBM provides a pre-compiled version of this program that is installed in the /usr/lpp/pkcs11/bin directory.

Enter OMVS and run the testpkcs11 utility, as shown in Example 3-15.

Example 3-15 Using testpkcs11 to validate PKCS#11 set-up

```
cd  /usr/lpp/pkcs11/bin
./testpkcs11 -t my.temp.token
Getting the PKCS11 function list...
Initializing the PKCS11 environment...
Creating the temporary token...
Opening a session...
Generating keys. This may take a while...
Enciphering data...
```

```
Deciphering data...
Destroying keys...
Closing the session...
Deleting the temporary token...
Test completed successfully§
```

Check for the message “Test completed successfully”, as shown in Example 3-15 on page 42.

3.4 Configuring a PKCS#11 token

PKCS#11 is a programming interface that is used to create and manipulate cryptographic tokens. The PKCS#11 tokens are containers that hold digital certificates and keys. IBM TouchToken for iOS components that run on z/OS use a PKCS#11 token to generate and manage secret keys and perform Hash-based Message Authentication Code (HMAC) operations.

PKCS #11 tokens and objects are stored in the TKDS. The TKDS serves as the repository for persistent cryptographic keys and certificates that are used by PKCS #11 applications.

Access to PKCS #11 tokens in ICSF is controlled by the CRYPTOZ class, with different access levels and a differentiation between standard users and security officers. For each token, the following resources in the CRYPTOZ class are used for controlling access to tokens:

- `USER.token-name` controls the access of the User role to the token.
- `SO.token-name` controls the access of the Security Officer (SO) role to the token.

z/OS PKCS #11 tokens are virtual and conceptually similar to RACF (SAF) key rings.

You can create and populate a PKCS #11 token in the following ways:

- By using the **`ADDTOKEN`** and **`BIND`** functions of the **`RACDCERT`** command.
- By using the `gskkyman` utility. For more information, see *z/OS Cryptographic Services System SSL Programming,* SC14-7495.
- Running an application that starts the PKCS #11 API that is provided by ICSF.

For more information about tokens and applications that use them, see *z/OS Cryptographic Services ICSF Writing PKCS #11 Applications,* SC14-7495.

We use the **`RACF RACDCERT`** command, which provides a familiar and easy-to-use interface. The TKDS definition is a prerequisite task.

3.4.1 Creating a CA keyring and certificate

Complete the following steps to create the needed keyring and certificates:

1. As shown in Example 3-16, create a CA keyring and certificate that is used to sign your registration server certificate. Then, export the certificate in binary mode to install it on your iPad or iPhone device.

Example 3-16 Generating and exporting the certificate

```
RACDCERT CERTAUTH GENCERT SUBJECT(CN('RACF CA for MFA server')  +
   OU('MFA TX9') O('IBM') ) +
   NOTAFTER(DATE(2025/12/31)) WITHLABEL('MFA TX9 RACF CA')
```

```
RACDCERT CERTAUTH LIST(LABEL('MFA TX9 RACF CA'))
RACDCERT CERTAUTH     EXPORT (LABEL('MFA TX9 RACF CA') ) +
   FORMAT (CERTDER) +
   DSN ('PRICHAR.MFA.CACERT.DER')

OPUT 'PRICHAR.MFA.CACERT.DER' '/u/prichar/mfacacert.der' binary

OSHELL chmod 744 /u/prichar/mfacacert.der
```

2. Create a server certificate for the registration server and connect it to the server keyring, as shown in Example 3-17.

Example 3-17 Creating server cetificate and conntecting it to server keyring

```
RACDCERT ID(AZFSTC) LIST(LABEL('MFA''s certificate'))
RACDCERT ID(AZFSTC) DELETE(LABEL('MFA''s certificate'))
RACDCERT ID(AZFSTC) LIST(LABEL('MFA''s certificate'))
RACDCERT GENCERT ID(AZFSTC) +
  SUBJECTSDN(CN('10.212.128.238')) +
  WITHLABEL('MFA''s certificate')  +
  SIZE(2048)   +
  NOTAFTER(DATE(2024/08/10))  +
  SIGNWITH(CERTAUTH LABEL('MFA TX9 RACF CA'))
RACDCERT ID(AZFSTC) LIST(LABEL('MFA''s certificate'))
```

Example 3-18 shows you how to use the **LIST** command with the certificate to check it.

Example 3-18 Listing the certificate

```
RACDCERT ID(AZFSTC) LIST(LABEL('MFA''s certificate'))
Digital certificate information for user AZFSTC:

  Label: MFA's certificate
  Certificate ID: 2QbB6cbi48PUxsF9okCDhZmjiYaJg4GjhUBA
  Status: TRUST
  Start Date: 2016/09/19 00:00:00
  End Date:   2024/08/10 23:59:59
  Serial Number:
       >01<
Issuer's Name:
        >CN=RACF CA for MFA server.OU=MFA TX9.O=IBM<
   Subject's Name:
        >CN=10.212.128.238<
   Signing Algorithm: sha256RSA
   Key Type: RSA
   Key Size: 2048
   Private Key: YES
   Ring Associations:
     Ring Owner: AZFSTC
     Ring:
        >mfa.server.KeyRing<
***
```

3. Create the a keyring for the registration server and connect the CERTAUTH certificate to this keyring, as shown in Example 3-19.

Example 3-19 Defining keyring for registartion server and connecting CERTAUTH to keyring

```
//STEP01 EXEC PGM=IKJEFT01
//SYSLBC DD DSN=SYS1.BRODCAST,DISP=SHR
//SYSPRINT  DD SYSOUT=*
//SYSTSPRT  DD SYSOUT=*
//SYSTSIN   DD *

  RACDCERT ID(AZFSTC) DELRING(mfa.server.KeyRing)
  RACDCERT ID(AZFSTC) ADDRING(mfa.server.KeyRing)

  RACDCERT ID(AZFSTC) CONNECT(CERTAUTH +
     LABEL('MFA TX9 RACF CA') +
     USAGE(CERTAUTH) RING(mfa.server.KeyRing)

  RACDCERT ID(AZFSTC) CONNECT(ID(AZFSTC) +
     LABEL('MFA''s certificate') +
     DEFAULT USAGE(PERSONAL) RING(mfa.server.KeyRing)

  RACDCERT ID(AZFSTC) LISTRING(mfa.server.KeyRing)
  RACDCERT ID(AZFSTC) LISTRING(*)

  RACDCERT ID(AZFSTC) LIST
```

4. List the keyring. It should contain the CA label (as CERTAUTH) and the server certificate label (as personal), as shown in Example 3-20.

Example 3-20 Listing the keyring

```
RACDCERT ID(AZFSTC) LISTRING(mfa.server.KeyRing)

Digital ring information for user AZFSTC:

   Ring:
        >mfa.server.KeyRing<
   Certificate Label Name              Cert Owner     USAGE      DEFAULT
   --------------------------------    ------------   --------   -------
   MFA TX9 RACF CA                     CERTAUTH       CERTAUTH     NO
   MFA's certificate                   ID(AZFSTC)     PERSONAL     YES
 ***
```

3.4.2 Creating the token

You can create your own PKCS#11 token by using the **RACDCERT ADDTOKEN** command , which the registration server then uses. You also can provide a valid token name and the registration server creates it if it does not exist. In either case, the token name you specify in this procedure must match the token name you use with AZFEXEC. As shown in Example 3-21 on page 46, we used the **RACDCERT ADDTOKEN** command in a batch job.

Example 3-21 Batch job step to create token by using RACDCERT

```
//ALTUSER  EXEC PGM=IKJEFT01,DYNAMNBR=30
//SYSLBC  DD DSN=SYS1.BRODCAST,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD  *
  RACDCERT ADDTOKEN(MFA.TOTP.TOKEN1)
/*
```

3.4.3 RACF controls for the token

Complete the following steps:

1. Activate the CRYPTOZ class with generics and RACLISTs.
2. Create a profile for the registration server's access to the token.
3. Create a profile for the standard user's access to the token.
4. Grant the registration server CONTROL access to the profile that protects the token, where `http-server-user-ID` is the user ID of the registration server started task.
5. Grant the same user UPDATE access to the profile that protects the token, where `http-server-user-ID` is the user ID of the registration server started task.
6. Refresh the profile for the CRYPTOZ class so that the changes take effect.

The command stream that can be used to set up the RACF controls is shown in Example 3-22.

Example 3-22 Establishing the RACF controls

```
/*                                                              */
/*Activate the CRYPTOZ class with generics and RACLISTs:        */
  SETROPTS CLASSACT(CRYPTOZ) GENERIC(CRYPTOZ) RACLIST(CRYPTOZ)
/*Create a profile for the registration server's access to token*/
  RDELETE CRYPTOZ SO.MFA.TOTP.TOKEN1
  RDEFINE CRYPTOZ SO.MFA.**    UACC(NONE)
/*Create a profile for the standard user's access to the token:  */
  RDELETE CRYPTOZ USER.MFA.TOTP.TOKEN1
  RDEFINE CRYPTOZ USER.MFA.**    UACC(NONE)
/*Give the registration server CONTROL access to the profile that */
/*protects the token, where http-server-user-ID */
/*is the user ID of the token, where http-server-user-ID */
  PERMIT SO.MFA.** CLASS(CRYPTOZ) ID(AZFSTC) ACC(CONTROL)
/*Give the same user UPDATE access to the profile that protects  */
/*token, where token, where http-server-user-ID */
/* of the registration server stc task. */
  PERMIT USER.MFA.** CLASS(CRYPTOZ) ID(AZFSTC) ACC(UPDATE)
/*Refresh the profile for the CRYPTOZ class, so that the changes */
/*effect:                                                        */
 SETROPTS RACLIST(CRYPTOZ) REFRESH
```

Permissions

The following accesses must be set up:

- Permit the user ID of MFA registration server to access the key ring by administering a profile in the FACILITY or the RDATALIB class.

- Permit the MFA ID read access to the IRR.DIGTCERT.LIST and IRR.DIGTCERT.LISTRING resources in the FACILITY class.
- Refresh the FACILITY general resource class.

A sample job extract to perform the set-up is shown in Example 3-23.

Example 3-23 Sample job step to set up accesses

```
//ALTUSER  EXEC PGM=IKJEFT01,DYNAMNBR=30
//SYSLBC  DD DSN=SYS1.BRODCAST,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD  *
 PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(AZFSTC) ACCESS(READ)
 PERMIT IRR.DIGTCERT.LIST CLASS(FACILITY) ID(AZFSTC) ACCESS(READ)
 SETROPTS RACLIST(FACILITY) REFRESH
```

3.5 Configuring the AT-TLS profile

Transport Layer Security (TLS) and Secure Sockets Layer (SSL) are protocol technologies that are used to protect data exchanges between client and server applications. By using TLS/SSL, the client and server can confirm the identity of each other and their data flows can be encrypted to protect e-commerce transactions and other sensitive data as they navigate across the network.

The z/OS Communications Server AT- TLS provides full transport layer security for all communication between the Apple device and the registration server. AT-TLS frees IBM TouchToken from having to be aware of the TLS details.

Configure an AT-TLS profile for HTTPS on the z/OS system you want to use as the IBM TouchToken registration server.

Note: Before the AT-TLS profile is configured, IBM z/OS Communications Server must be installed and configured. Also, you must be familiar with AT- TLS policies.

Our installation procedure assumes that a public certificate authority (CA) is not used. We defined our own CA because we did not want to request that our server certificate be signed by a well-know commercial CA.

It is recommended that your IBM TouchToken registration server use a certificate that is issued by a well-known CA. Because we are not using a CA that is trusted by default by Apple iOS, we must ensure that all IBM TouchToken for iOS devices include a Configuration Profile that us installed that allows them to establish TLS connections with the registration server.

We describe how to define a new configuration profile on your iOS device that allows the use of the CA that we defined.

Note: If your registration server certificate was not issued by a well-known CA, do not instruct users to visit the registration server start page until a Configuration Profile is installed with which they can establish TLS connections with the registration server.

If users accept the registration server certificate in Mobile Safari as an SSL exception, the IBM TouchToken for iOS application still cannot trust the CA that issued the certificate. Users can view the enrollment URL, but they cannot complete the enrollment process.

The MFA registration server acts as an “imbedded” Apache web server. It allows secured SSL connections only. However, this registration server cannot be customized and accepts no configuration parameters. Therefore, you cannot specify directives, such as key ring name, list of cipher suites supported, and encryption protocol.

When a user enrolls a new IBM TouchToken account by using the IBM TouchToken for iOS application, sensitive data flows from the registration server to the application that is running on the user’s iOS device. HTTPS must be used to protect that data. The TLS configuration that is used by the registration server must be compatible with Application Transport Security policy as enforced by Apple iOS.

z/OS provides the AT/TLS function, which provides encapsulation of the SSL protocol for any application that cannot implement the SSL/TLS protocol alone.

We use that support to control MFA secure connections. The AT-TLS protection is driven by the security policies you choose for the application. Policy Agent (PAGENT) is a system component that reads your security policies and interfaces with the TCP/IP stack to enforce them. Therefore, we must create and activate an AT/TLS policy.

AT-TLS policy conditions consist of various selection criteria that act as filters for AT-TLS rules. Traffic can be filtered based on local addresses, remote addresses, local port range, remote port range, job name, user identification, and direction.

TCP/IP and PAGENT then ensure that the iOS device and the registration server identify and authenticate each other by using SSL handshakes before establishing a connection. They encrypt the data that is exchanged between them.

If you are not familiar with ATTLS, see the following IBM publications:

- *z/OS Communications Server: IP Configuration Guide*, SC27-3650
- *z/OS Communications Server: IP Configuration Reference*, SC27-3651

For more information, see *IBM z/OS V2R1 Communications Server TCP/IP Implementation Volume 4: Security and Policy-Based Networking*, SG24-8099, which is available at this website:

`http://www.redbooks.ibm.com/abstracts/sg248099.html`

The following overall process is used to create an AT-TLS policy:

1. Set up the PAGENT started task.
2. Create digital certificates and key rings, as described in 3.4.1, “Creating a CA keyring and certificate” on page 43.
3. Configure AT-TLS policy for MFA registration server and iOS client and create Policy Agent files:
   a. Create a Policy Agent main configuration file that contains a TcpImage statement for the server stack.
   b. Create a Policy Agent image configuration file for the server stack.
   c. Create image-specific AT-TLS configuration files and (optionally) common AT-TLS configuration files on the policy server.
4. Add AT-TLS configuration. For local AT-TLS policies, add a TTLSConfig statement to the Policy Agent image configuration file, which identifies the following TTLSConfig policy file location:

   ```
   TTLSConfig serverpath
   ```

5. Add the AT-TLS policy statements to the serverpath file.

6. Update necessary RACF profiles and set up InitStack access control:
   a. Define the `EZB.INITSTACK.sysname.tcpname` profile for each AT-TLS stack.
   b. Permit administrative applications to use the stack before AT-TLS is initialized.
7. Enable AT-TLS. Set TCPCONFIG TTLS in `PROFILE.TCPIP`.

The z/OSMF Configuration Assistant provides a simplified method of generating TCP/IP networking policies.

You can use the z/OSMF Configuration Assistant to create configuration files for any number of z/OS images with any number of TCP/IP stacks per image. The Configuration Assistant reduces configuration complexity by providing a consistent and easily manageable interface. It can dramatically reduce the amount of time that is required to generate and maintain policy files for Communications Server disciplines. The Configuration Assistant is intended to replace manual configuration of the policy disciplines, but it can also incorporate policy data directly from the Policy Agent.

Application Transparent TLS was introduced by z/OS Communications Server to provide a way to secure an application by using industry-standard Transport Layer Security without introducing application-specific code and configuration controls directly into the application.

Note: If PAGENT and AT-TLS is not implemented in your own installation, see *RACF Remote Sharing Facility over TCP/IP,* SG24-8041, which provides step-by-step procedures to implement TCPIP PAGENT and the AT-TLS policy through the z/OSMF configuration assistant. It also shows this process for another TCPIP application (RRSF), but still provides invaluable help and can serve as a model for defining your own AT-TLS policy for MFA.

You must make the necessary changes regarding the port number that is used (the MFA example uses port 6789, but you can choose your own value). You must specify this port when you run AZFEXEC to configure the registration server later.

You must change the specific set of ciphers to be compatible with the Apple Touch ID device, and the name of the SAF key ring we created.

You must use the z/OSMF Configuration Assistant to Configure MFA registration server SSL traffic with AT-TLS.

The following areas must be addressed when z/OSMF is used:

- Create self-signed root certificate in RACF.
- Create server key ring and certificate that is signed by the root certificate.
- Define ATTLS policy with the z/OSMF Configuration Assistant.
- Transfer policy definition file to z/OS and enable Policy Agent.

Complete the following steps to prepare the environment and use the z/OSMF configuration assistant:

1. Before starting the PAGENT, create the appropriate RACF definitions to set up the InitStack access control:
   a. Define the `EZB.INITSTACK.sysname.tcpname` profile for each AT-TLS stack.
   b. Permit administrative applications to use the stack before AT-TLS is initialized. For examples of the security product commands that are needed to create this resource profile name and grant users access to it, see member EZARACF in sample data set SEZAINST.

Example 3-24 shows how this set up might be achieved.

Example 3-24 PAGENT RACF set up and access

```
//TSJOOOA JOB 3OOOOOOO,'MFA JOB CARD ',MSGLEVEL=(1,1),
// CLASS=A,MSGCLASS=Q,NOTIFY=&SYSUID,REGION=OM
//*****************************************************************
//*
//* If you chose not to use the RACF STARTED Class,
//* the entry should be placed in the started procedures table ICHRINO3
//*
//PAGENT EXEC PGM=IKJEFTO1
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
ADDUSER PAGENT DFLTGRP(SYS1) OMVS(UID(0) HOME('/'))
SETROPTS CLASSACT(STARTED)
SETROPTS RACLIST(STARTED)
SETROPTS GENERIC(STARTED)
RDEFINE STARTED PAGENT.* STDATA(USER(PAGENT))
SETROPTS RACLIST(STARTED) REFRESH
SETROPTS GENERIC(STARTED) REFRESH
//*
//*** PERMIT Pagent to the BPX.DAEMON Facility
//*
//*
//BPXPAGNT EXEC PGM=IKJEFTO1
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
SETROPTS CLASSACT(SERVAUTH)
SETROPTS RACLIST (SERVAUTH)
SETROPTS GENERIC (SERVAUTH)
SETROPTS CLASSACT(FACILITY)
SETROPTS RACLIST (FACILITY)
RDEFINE FACILITY BPX.DAEMON UACC(NONE)
PERMIT BPX.DAEMON CLASS(FACILITY) ID(PAGENT) ACCESS(READ)
SETROPTS RACLIST(FACILITY) REFRESH
RDEFINE SERVAUTH EZB.PAGENT.** UACC(NONE)
PERMIT EZB.PAGENT.** CLASS(SERVAUTH) -
ID(PAGENT) ACCESS(READ)
PERMIT EZB.PAGENT.** CLASS(SERVAUTH) -
ID(SYS1) ACCESS(READ)
PERMIT EZB.PAGENT.** CLASS(SERVAUTH) -
ID(SYS1) ACCESS(READ)
SETROPTS GENERIC(SERVAUTH) REFRESH
SETROPTS RACLIST(SERVAUTH) REFRESH
/*
//PAGENT EXEC PGM=IKJEFTO1
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
RDEFINE SERVAUTH EZB.INITSTACK.*.* UACC(NONE)
PERMIT EZB.INITSTACK.*.* CLASS(SERVAUTH) ID(CFZSRVGP) ACCESS(READ)
PERMIT EZB.INITSTACK.*.* CLASS(SERVAUTH) ID(OMVSGRP) ACCESS(READ)
PERMIT EZB.INITSTACK.*.* CLASS(SERVAUTH) ID(WSCFG1) ACCESS(READ)
PERMIT EZB.INITSTACK.*.* CLASS(SERVAUTH) ID(SYS1) ACCESS(READ)
SETROPTS GENERIC(SERVAUTH) REFRESH
SETROPTS RACLIST(SERVAUTH) REFRESH
RLIST SERVAUTH EZB.PAGENT.** AUTHUSER
/*
```

2. Set up the PAGENT started task procedure. A sample is included in `TCPIP.SEZAINST(EZAPAGSP)`. The PAGENT started task procedure that we used is shown in Example 3-25.

Example 3-25 PAGENT started task procedure

```
//PAGENT   PROC
//*
//* IBM Communications Server for z/OS
//* SMP/E distribution name: EZAPAGSP
//*
//* 5694-A01 Copyright IBM Corp. 1998, 2007
//* Licensed Materials - Property of IBM
//* "Restricted Materials of IBM"
//* Status = CSV1R9
//*
//PAGENT   EXEC PGM=PAGENT,REGION=0K,TIME=NOLIMIT,
//       PARM='ENVAR("_CEE_ENVFILE=DD:STDENV")/'
//STDENV   DD DSN=TCPIP.TCPPARMS(PAGENT),DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//*
//CEEDUMP  DD SYSOUT=*,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)
```

The PAGENT started task procedure includes an STDENV DD, which points to the data set that contains the TCIP parameters to be used. The parameters are shown in Example 3-26.

Example 3-26 PAGENT TCP/IP parameters

```
PAGENT_CONFIG_FILE=/etc/pagent.conf
PAGENT_LOG_FILE=/tmp/pagent.log
LIBPATH=/usr/lib
TZ=GMT-1
```

3. Enable AT-TLS in the TCP/IP profile. To activate AT-TLS, the TTLS parameter must be added to the TCPCONFIG profile statement in each stack that is to support AT-TLS, as shown in Example 3-27.

Example 3-27 TCPPARMS profile showing TTLS parameter

```
TCPCONFIG
          INTERVAL            10     ; defaultkeepalive (10=default)
          RESTRICTLOWPORTS           ; reserve low ports for server
          TCPSENDBFRSIZE      128K   ; 256 to 256K dft  16K
          TCPRCVBUFRSIZE      128K   ; 256 to 256K dft  16K
          TCPMAXRCVBUFRSIZE   512K   ; max 512 K   dft  256K
          SENDGARBAGE         FALSE  ; default value
          NODELAYACKS
          FINWAIT2TIME        60
          TTLS                       ; AT-TLS support:
```

4. To activate the profile, you can recycle TCP/IP or use an **`OBEYFILE`** command if you want do activate it dynamically. (Check your site's practices and procedures for recycling TCP/IP.)
5. Protect the TCP/IP stack. The TCP/IP stack is protected by a SERVAUTH class profile that uses the format `EZB.STACKACCESS.sysname.tcpstackname`.

You can use the sample commands that are shown in Example 3-28 to protect RACF and to grant RACF User ID access to it. For examples of the security product commands that are needed to create this resource profile name and grant users access to it, see member EZARACF in sample data set SEZAINST.

Example 3-28 Assigning RACF User ID

```
RDEFINE SERVAUTH EZB.STACKACCESS.*.* UACC(NONE)
PERMIT EZB.STACKACCESS.*.* CLASS(SERVAUTH) ID(RACF) ACCESS(READ)
SETROPTS RACLIST(SERVAUTH) REFRESH
```

6. (Optional) Define and protect the MFA port 6789 (or the port that you chose). The updated statements to define the MFA port in the TCP/IP profile are shown in Example 3-29.

Example 3-29 Defining the MFA port

```
;;-------------------------------------------------------
;; MFA port could be reserved. The default port value is 6789
PORT
6789 TCP AZF#IN01 ; MFA registration listener port
;;-------------------------------------------------------
```

(Alternative) You can instead use profile `EZB.PORTACCESS.*.*` in class SERVAUTH to protect the MFA TCP/IP port and gave MFA RACF User ID access to it. You specify the SAF keyword and User ID AZFSTC to protect the port by using a RACF profile.

7. Start the PAGENT started task procedure by issuing the MVS **`S PAGENT`** command and the messages that are shown in Example 3-30 should be issued.

Example 3-30 PAGENT initial start messages

```
EZZ8431I PAGENT STARTING
EZZ8432I PAGENT INITIALIZATION COMPLETE
EZZ4249I TCPIP INSTALLED TTLS POLICY HAS NO RULES
EZD1586I PAGENT HAS INSTALLED ALL LOCAL POLICIES FOR TCPIP
EZD1576I PAGENT IS READY FOR SERVICES CONNECTION REQUESTS
```

The message EZZ4249I indicates that there are no TTLS rules.

Consider the following basic concepts for defining AT-TLS policies:

- Traffic Descriptor:
  - Identifies certain types of traffic by using transport protocol (TCP or UDP), local and remote port numbers, for TCP the direction of the connection setup, and so on.
  - Specific AT-TLS attributes for this type of application.
  - A traffic descriptor does not include references to IP addresses in any form.
- Security Level: Identifies the SSL/TLS security requirements, such as cipher suites and allowed protocol versions (SSLv2, SSLv3, TLSv1).
- Requirement Map: Links traffic descriptors to security levels.

8. Configure TTLS rules by using the z/OSMF configuration assistant. Log on to z/OSMF with an administrator ID. Under Configuration tab, click **Configuration Assistant** task.

We do not describe all aspects of a Configuration Assistant dialog here. Instead, we focus in the following steps on how to configure AT-TLS policies:

a. At the end of the process after you create the policy, you must transfer the policy flat file to z/OS. Expand the arrow that is to the right of your stack TCPIP and select **Install configuration files**.

b. Enter your policy file's name (`attls-pol`) and location (`/etc`).

c. Click **Select Actions** → **Install**.

d. Select the **FTP** option and enter your transfer information.

9. After the transfer completes, switch to your PCOM session. Then, go to the ISHELL or UDLIST and browse the content of your AT-TLS policy by using the following command:

```
BROWSE    /X9/etc/attls-pol
```

The policy is shown in Example 3-31.

Example 3-31 AT-TLS policy

```
##
## AT-TLS Policy Agent Configuration file for:
##    Image: TX9
##    Stack: TCPIP
##
## Created by the IBM Configuration Assistant for z/OS Communications Server
## Version 1 Release 10
## Backing Store = C:\...\IBM\zCSConfigAssist\V1R10\files\tx9initialconfig
## FTP History:
## 2010-02-04 17:07:46  prichar to tx9
## 2010-02-04 15:39:17  prichar to tx9
## 2010-02-04 15:19:16  prichar to tx9
##
## End of Configuration Assistant information
TTLSRule ttls_AZF
{
   LocalAddr ALL
   RemoteAddr ALL
   LocalPortRange 6789 <=== See note A
   Direction Inbound
   Priority 255
   TTLSGroupActionRef gAct1~AZF
   TTLSEnvironmentActionRef eAct1~AZF
   TTLSConnectionActionRef cAct1~AZF
}
TTLSGroupAction gAct1~AZF
{
   TTLSEnabled On
   Trace 5
}
TTLSEnvironmentAction eAct1~AZF
{
   HandshakeRole Server
   EnvironmentUserInstance 0
   TTLSEnvironmentAdvancedParmsRef eAdv1~AZF
   TTLSKeyringParmsRef keyR1~AZF        <===See note B
   Trace 5
}
TTLSConnectionAction cAct1~AZF
{
   HandshakeRole Server
   TTLSCipherParmsRef cipher-AZF      <=== See note C
   TTLSConnectionAdvancedParmsRef cAdv1~AZF
   CtraceClearText Off
```

```
   Trace 5
}
TTLSConnectionAdvancedParms cAdv1~AZF
{
   ApplicationControlled Off
   SecondaryMap Off
}
TTLSEnvironmentAdvancedParms eAdv1~AZF
{
   ApplicationControlled Off
   SSLv2 Off
   SSLv3 Off
   TLSv1 Off      <=== see note D
   TLSv1.1 On
   TLSv1.2 On
}
TTLSKeyringParms keyR1~AZF
{
   Keyring mfa.server.KeyRing c)
}
TTLSCipherParms cipher-AZF     b)
{
   V3CipherSuites TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
   V3CipherSuites TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
   V3CipherSuites TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
   V3CipherSuites TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
   V3CipherSuites TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
   V3CipherSuites TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
   V3CipherSuites TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
   V3CipherSuites TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
}
```

Consider the following notes as highlighted in Example 3-31 on page 53:

- Note A: The example uses port 6789, but you can choose your own value. This port must be specified when you run AZFEXEC to configure the registration server.
- Note B: The name of the SAF key ring that you created.
- Note C: Use this specific set of ciphers to be compatible with the Apple Touch ID device.
- Note D: We suggest sure you include TLSv1 Off; otherwise, problems can occur. The TTLS policy is failing because many browsers now do not accept the TTLS 1.0 protocol. You must remove the TTLS 1.0 (off).

The symptom that you might receive when you attempt to connect to the registration server by using a browser is shown in Example 3-32.

Example 3-32 TTLS 1.0 problem symptoms

```
https://10.212.128.238:6789/AZFTOTP1/start
>>Secure Connection Failed

An error occurred during a connection to xxx Cannot communicate securely with
peer: no common encryption algorithm(s). (Error code:
ssl_error_no_cypher_overlap)
```

```
   The page you are trying to view cannot be shown because the authenticity of
the received data could not be verified.
   Please contact the website owners to inform them of this problem.<<

In the SYSLOG, you will see the following messages:
EZD1287I TTLS Error RC: 6013 Data Decryption 962
  LOCAL: 10.212.128.238..6789
  REMOTE: 10.212.131.34..57509
  JOBNAME: AZF#INO1 RULE: ttls_AZF
  USERID: AZFSTC GRPID: 00000003 ENVID: 0000000D CONNID: 0000BA2C
```

10. Load and start the ATTLS policy with policy agent. Go to the OMVS shell, and enter the shell commands that are shown in Example 3-33.

Example 3-33 Load and start the policy agent

```
pasearch -p TCPIP
PRICHAR@X9:/u/prichar: #pasearch -p TCPIP
No policies retrieved
```

11. Return to ishell. Edit the PAGENT configuration file (`/etc/pagent.conf`), and specify the ATTLS policy file that we created (`/etc/attls-pol`). The display should look similar to Example 3-34.

Example 3-34 Adding the policy to the pagent.conf file

```
## /etc/pagent.conf
##
## Image: x9
##
ServicesConnection
TcpImage TCPIP FLUSH 600
##
TTLSConfig /etc/attls-pol FLUSH NOPURGE
```

12. Save the member. Run the **F PAGENT,REFRESH** system command to refresh the PAGENT configuration. The responses to the command are shown in Example 3-35.

Example 3-35 Refreshing the PAGENT address space

```
F PAGENT,REFRESH
EZZ8443I PAGENT MODIFY COMMAND ACCEPTED
EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIP : TTLS
EZD1289I TCPIP ICSF SERVICES ARE CURRENTLY AVAILABLE FOR AT-TLS GROUP
gAct1 AZF
```

13. Return to the OMVS shell and enter the commands that are shown in Example 3-36. The responses to the commands also are shown.

Example 3-36 Shell commands and responses to show policy information

```
pasearch -c -t TCPIP
UserID:/u/prichar #pasearch -c -t TCPIP

TCP/IP pasearch CS V2R2                  Image Name: TCPIP
  Date:                 10/18/2016       Time:  10:10:22
  PAPI Version:         12               DLL Version:  12
```

```
TTLS Policy Object:
  ConfigLocation:       Local             LDAPServer:        False
  CommonFileName:
  ImageFileName:        /etc/attls-pol
  ApplyFlush:           True              PolicyFlush:       True
  ApplyPurge:           True              PurgePolicies:     False
  AtomicParse:          True              DeleteOnNoflush:   False
  DummyOnEmptyPolicy:   True              ModifyOnIDChange:  False
  Configured:           True              UpdateInterval:    600
  TTLS Enabled:         True
  InstanceId:           1474359797
  LastPolicyChanged:    Tue Sep 20 09:23:17 2016
PRICHAR@X9:/u/prichar #
```

pasearch -p TCPIP

```
UserID:/u/prichar #pasearch -p TCPIP
TCP/IP pasearch CS V2R2                   Image Name: TCPIP
  Date:                 10/18/2016        Time:  10:14:07
  TTLS Instance ID:     1474359797

policyRule:             ttls_AZF
  Rule Type:            TTLS
  Version:              3                 Status:            Active
  Weight:               255               ForLoadDist:       False
  Priority:             255               Sequence Actions:  Don't Care
  No. Policy Action:    3
  policyAction:         gAct1~AZF
   ActionType:          TTLS Group
   Action Sequence:     0
  policyAction:         eAct1~AZF
   ActionType:          TTLS Environment
   Action Sequence:     0
  policyAction:         cAct1~AZF
   ActionType:          TTLS Connection
   Action Sequence:     0
  Time Periods:
   Day of Month Mask:
   First to Last:       11111111111111111111111111111111
   Last to First:       11111111111111111111111111111111
Month of Yr Mask:    111111111111
  Day of Week Mask:    1111111  (Sunday - Saturday)
  Start Date Time:     None
  End Date Time:       None
  Fr TimeOfDay:        00:00             To TimeOfDay:      24:00
  Fr TimeOfDay UTC:    22:00             To TimeOfDay UTC:  22:00
  TimeZone:            Local
 TTLS Condition Summary:                 NegativeIndicator: Off
  Local Address:
   FromAddr:           All
   ToAddr:             All
  Remote Address:
   FromAddr:           All
   ToAddr:             All
  LocalPortFrom:       6789              LocalPortTo:       6789
  RemotePortFrom:      0                 RemotePortTo:      0
```

```
  JobName:                                UserId:
  ServiceDirection:    Inbound
 Policy created: Tue Sep 20 09:23:17 2016
 Policy updated: Tue Sep 20 09:23:17 2016

 TTLS Action:                  gAct1~AZF
   Version:                    3
   Status:                     Active
Scope:                       Group
    TTLSEnabled:                On
    CtraceClearText:            Off
    Trace:                      5
    FIPS140:                    Off
    TTLSGroupAdvancedParms:
     SecondaryMap:              Off
     SyslogFacility:            Daemon
    Policy created: Tue Sep 20 09:23:17 2016
    Policy updated: Tue Sep 20 09:23:17 2016

  TTLS Action:                  eAct1~AZF
    Version:                    3
    Status:                     Active
    Scope:                      Environment
    HandshakeRole:              Server
    Trace:                      5
    SuiteBProfile:              Off
    TTLSKeyringParms:
     Keyring:                   mfa.server.KeyRing
    TTLSEnvironmentAdvancedParms:
     SSLv2:                     Off
     SSLv3:                     Off
     TLSv1:                     Off
TLSv1.1:                    On
     TLSv1.2:                   On
     ApplicationControlled:     Off
     HandshakeTimeout:          10
     ClientAuthType:            Required
     ResetCipherTimer:          0
     TruncatedHMAC:             Off
     CertValidationMode:        Any
     ServerMaxSSLFragment:      Off
     ClientMaxSSLFragment:      Off
     ServerHandshakeSNI:        Off
     ClientHandshakeSNI:        Off
     Renegotiation:             Default
     RenegotiationIndicator:    Optional
     RenegotiationCertCheck:    Off
    TTLSGskAdvancedParms:
     TTLSGskHttpCdpParms:
      HttpCdpEnable:            Off
      HttpCdpProxyServerPort:   80
      HttpCdpResponseTimeout:   15
      HttpCdpMaxResponseSize:   204800
      HttpCdpCacheSize:         32
      HttpCdpCacheEntryMaxsize: 0
```

```
     TTLSGskOcspParms:
OcspAiaEnable:            Off
      OcspProxyServerPort:      80
      OcspRetrieveViaGet:       Off
      OcspUrlPriority:          On
      OcspRequestSigalg:
       0401  TLS_SIGALG_SHA256_WITH_RSA
      OcspClientCacheSize:      256
      OcspCliCacheEntryMaxsize: 0
      OcspNonceGenEnable:       Off
      OcspNonceCheckEnable:     Off
      OcspNonceSize:            8
      OcspResponseTimeout:      15
      OcspMaxResponseSize:      20480
    EnvironmentUserInstance:    0
    Policy created: Tue Sep 20 09:23:17 2016
    Policy updated: Tue Sep 20 09:23:17 2016

  TTLS Action:                  cAct1~AZF
    Version:                    3
    Status:                     Active
    Scope:                      Connection
    HandshakeRole:              Server
    CtraceClearText:            Off
    Trace:                      5
TTLSConnectionAdvancedParms:
     SecondaryMap:              Off
     ApplicationControlled:     Off
    TTLSCipherParms:
     v3CipherSuites:
      C027  TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
      C028  TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
      C02F  TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
      C030  TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
    Policy created: Tue Sep 20 09:23:17 2016
    Policy updated: Tue Sep 20 09:23:17 2016
```

3.6 Testing the TLS configuration

Complete the following steps to test the new TLS configuration:

1. Start the registration server. The command and response are shown in Example 3-37.

Example 3-37 Starting the registration server AZF#IN01

```
S AZF#IN01
$HASP100 AZF#IN01 ON STCINRDR
IEF695I START AZF#IN01 WITH JOBNAME AZF#IN01 IS ASSIGNED TO USER AZFSTC
 , GROUP AZFGRP
$HASP373 AZF#IN01 STARTED
IEF403I AZF#IN01 - STARTED - TIME=10.47.04
```

In the AZF#IN01 server log, look for the messages that are shown in Example 3-38.

Example 3-38 AZF#IN01 server log

```
AZFTOTPWEB:AZF5050I Console listener task starting up
AZFTOTPWEB:AZF5002I Server base init success (port 6789)
AZFTOTPWEB:constructed JSON to embed in launchURL:
{"reqtype":"regStart","realmN
AZFTOTPWEB:b64EncJson:
eyJyZXFOeXBlIjoicmVnU3RhcnQiLCJyZWFsbU5hbWUiOiJUWDlUT1RQI
AZFTOTPWEB:launchURL:
azftotp://eyJyZXFOeXBlIjoicmVnU3RhcnQiLCJyZWFsbU5hbWUiOiJU
121 reason 0x76650446
```

2. Open a web browser at your MFA registration server IP address and port by using HTTPS, as shown in Example 3-39.

Example 3-39 Pointing the web browser to your MFA registration server

```
https://10.212.128.238:6789/AZFTOTP1/start
```

Your own IP address is assigned. The page that is shown in Figure 3-1 should be displayed.

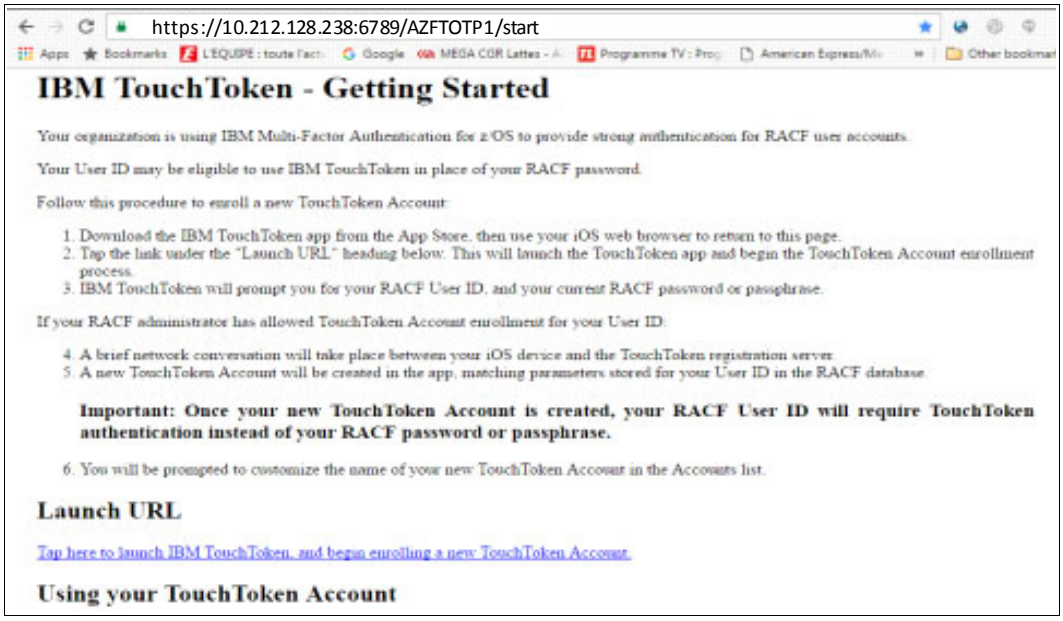

Figure 3-1 IBM TouchToken Getting Started page

Note: The IBM TouchToken application is not meant to be used on a workstation. The registration process is supported on iOS devices only. However, opening the Welcome page allows us to determine whether the AT-TLS policy was configured correctly and if it is active.

3. Enter the MVS Display command that is shown in Example 3-40 to show the TTLS connections. The response to the command is also shown in Example 3-40.

Example 3-40 Display the TTLS connections

```
D TCPIP,,N,ALLC,CONNT=TTLSP
EZD0101I NETSTAT CS V2R2 TCPIP 832
USER ID  CONN     STATE
AZF#IN01 0017BE04 ESTBLSH
  LOCAL SOCKET:   10.212.128.238..6789
  FOREIGN SOCKET: 10.81.120.5..57825
```

The report shows that an active connection 17BE04 is present. Use the MVS command to Display connection information that relates to 17BE04, as shown in Example 3-41.

Example 3-41 Display information about connection 17BE04

```
D TCPIP,,N,TTLS,CON=17BE04
EZD0101I NETSTAT CS V2R2 TCPIP 841
CONNID: 0017BE04
  JOBNAME:      AZF#IN01
  LOCALSOCKET:  10.212.128.238..6789
  REMOTESOCKET: 10.81.120.5..57828
  SECLEVEL:     TLS VERSION 1.2
  CIPHER:       C02F TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
  CERTUSERID:   N/A
  MAPTYPE:      PRIMARY
  FIPS140:      OFF
  SESSIONID:    000100C7 09517805 DF160000 00000000
                00000000 00000000 5805F0B6 0000003E
  SIDREUSEREQ:  OFF
TTLSRULE: TTLS_AZF
  TTLSGRPACTION:  GACT1 AZF
  TTLSENVACTION:  EACT1 AZF
  TTLSCONNACTION: CACT1 AZF
1 OF 1 RECORDS DISPLAYED
END OF THE REPORT
```

From the display response, you can determine whether ATTLS is active and which rule was applied for your HTTPS connection (cipher suite, TLS version, and so on).

You also can use the TSO NETSTAT commands, as shown in Example 3-42.

Example 3-42 Display information that uses the TSO NETSTAT command

```
NETSTAT ALLCONN APPLDATA TCP TCPIP ( CONNT TTLSP
MVS TCP/IP NETSTAT CS V2R2       TCPIP Name: TCPIP           10:11:10
User ID  Conn     State
-------  ----     -----
AZF#IN01 00026DE9 Establish
  Local Socket:   10.212.128.238..6789
  Foreign Socket: 10.212.131.34..54851

NETSTAT TTLS CO 00026DE9 DETAIL tcp tcpip
MVS TCP/IP NETSTAT CS V2R2       TCPIP Name: TCPIP           10:12:32
ConnID: 00026DE9
  JobName:      AZF#IN01
  LocalSocket: 10.212.128.238..6789
```

```
  RemoteSocket: 10.212.131.34..54851
  SecLevel:     TLS Version 1.2
  Cipher:       C02F TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
  CertUserID:   N/A
  MapType:      Primary
  FIPS140:      Off
  SessionID:    000100C7 09D48322 D6430000 00000000
                00000000 00000000 57E3AD6B 00000013
  SIDReuseReq:  Off
TTLSRule: ttls_AZF
  Priority:       255
  LocalAddr:      All
  LocalPort:      6789
  RemoteAddr:     All
  RemotePort:     All
  Direction:      Inbound
  TTLSGrpAction:  gAct1:AZF
    GroupID:                    00000009
    TTLSEnabled:                On
    CtraceClearText:            Off
    Trace:                      5
    SyslogFacility:             Daemon
    SecondaryMap:               Off
    FIPS140:                    Off
  TTLSEnvAction:  eAct1:AZF
    EnvironmentUserInstance:    0
    HandshakeRole:              Server
    SuiteBProfile:              Off
    Keyring:                    mfa.server.KeyRing
    Trace:                      5
    SSLV2:                      Off
    SSLV3:                      Off
    TLSV1:                      Off
    TLSV1.1:                    On
    TLSV1.2:                    On
    ResetCipherTimer:           0
    ApplicationControlled:      Off
     HandshakeTimeout:           10
     TruncatedHMAC:              Off
     ClientMaxSSLFragment:       Off
     ServerMaxSSLFragment:       Off
     ClientHandshakeSNI:         Off
     ServerHandshakeSNI:         Off
     ClientAuthType:             Required
     CertValidationMode:         Any
     Renegotiation:              Default
     RenegotiationIndicator:     Optional
     RenegotiationCertCheck:     Off
     HttpCdpEnable:              Off
     HttpCdpProxyServerPort:     80
     HttpCdpResponseTimeout:     15
     HttpCdpMaxResponseSize:     204800
     HttpCdpCacheSize:           32
     HttpCdpCacheEntryMaxsize:   0
     OcspAiaEnable:              Off
```

```
    OcspProxyServerPort:        80
    OcspRetrieveViaGet:         Off
    OcspUrlPriority:            On
    OcspRequestSigalg:          0401 TLS_SIGALG_SHA256_WITH_RSA
    OcspClientCacheSize:        256
    OcspCliCacheEntryMaxsize:   0
    OcspNonceGenEnable:         Off
    OcspNonceCheckEnable:       Off
    OcspNonceSize:              8
    OcspResponseTimeout:        15
    OcspMaxResponseSize:        20480
  TTLSConnAction: cAct1:AZF
    HandshakeRole:              Server
    V3CipherSuites:             C027 TLS_ECDHE_RSA_WITH_AES_128_CB
                                     C_SHA256
                                C028 TLS_ECDHE_RSA_WITH_AES_256_CB
                                     C_SHA384
                                C02F TLS_ECDHE_RSA_WITH_AES_128_GC
                                     M_SHA256
                                C030 TLS_ECDHE_RSA_WITH_AES_256_GC
                                     M_SHA384
    CtraceClearText:            Off
    Trace:                      5
    ApplicationControlled:      Off
    SecondaryMap:               Off
***
```

You now successfully used the Configuration Assistant feature of z/OSMF to deploy an ATTLS TCPIP policy on your z/OS TCPIP stack.

The Configuration Assistant is intended to replace manual configuration of the policy disciplines, but it can also incorporate policy data directly from the Policy Agent.

You also successfully tested your AT-TLS policy against your MFA registration server. The registration server can be stopped by using the **P AZF#IN01** command.

4

Configuring IBM TouchToken

In this chapter, we describe how to configure the IBM TouchToken support to make it ready for use.

This chapter includes the following topics:

4.1 Time-based One-time Password algorithm

Time-based One-time Password (TOTP) is an algorithm that computes a one-time password from a shared secret key and the current time. The seconds that are used to calculate the time-based TOTP seed are always in UTC (time zone +0:00) time. TOTP is an example of a hash-based message authentication code (HMAC). It combines a secret key with the current time stamp by using a cryptographic hash function to generate a one-time password. Because the time stamp typically increases in 30-second intervals, passwords that are generated close together in time from the same secret key are equal.

In a typical two-factor authentication application, user authentication occurs by using the following process:

- A user enters a user name and password into a website or other server, which generates a one-time password for the server that uses TOTP that is running locally on a smartphone or similar device.
- The user also enters the generated password into the server.
- The server also runs TOTP to verify the entered one-time password. For this verification to work, the clocks of the user's device and the server must be roughly synchronized. The server typically accepts one-time passwords that are generated from timestamps that differ by ±1 time interval from the client's time stamp.

A single secret key, which is to be used for all subsequent authentication sessions, must be shared between the server and the user's device over a secure channel in advance.

The Internet Engineering Task Force (IETF) request for comments (rfc) 6238 (which is related to TOTP) states:

> *The prover (for example, token or soft token) and verifier (authentication or validation server) must know or derive the current UNIX time (that is, the number of seconds that elapsed since midnight UTC of January 1, 1970) for OTP generation. The precision of the time that is used by the prover affects how often the clock synchronization must occur.*

If your host system and your iOS device do not have the same GMT time, ensure that you allow for this time gap between the two peers based on their time difference.

Configuring IBM TouchToken requires completed tasks on the host and smart device.

4.2 Configuring on z/OS

You must now configure the IBM TouchToken registration server settings.

The PKCS#11 tokens and Application Transparent Transport Layer Security (AT-TLS) must be configured before you configure the IBM TouchToken plug-in. Complete the following steps:

1. Log on to TSO. On the ISPF Primary menu, select **Option 6: TSO command**.
2. Check that you allocated the AZF.* data sets. Ensure that you copied the AZF.SAZFECEC(AZFEXEC) into your SYSEXEC concatenation. The AZFEEC dynamically allocates the ISPF application data sets.
3. Run the REXX procedure AZFEXEC, as shown in Example 4-1 on page 65.

Example 4-1 Running the MFA AZFEXEC procedure

```
===> AZFEXEC
```

The IBM Multi-Factor Authentication for z/OS Plug-in Administration ISPF panel opens. Select the **AZFTOTP1 Plug-in** option by entering `T` on the command line, as shown in Figure 4-1.

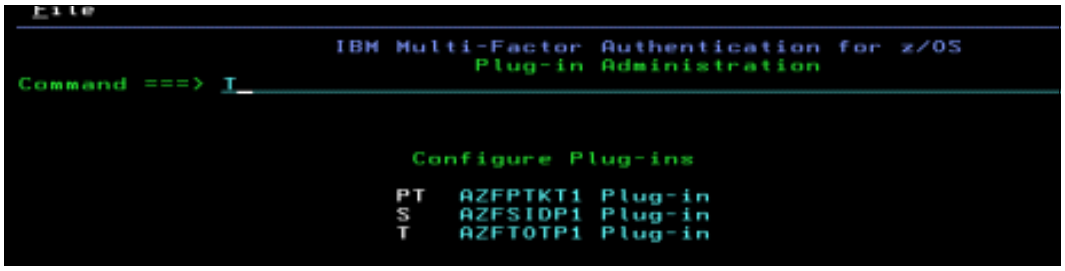

Figure 4-1 Selecting the AZFTOTP1 option

By using the T option, you configure the AZFTOTP plug-in. Ensure that you have the following RACF attributes to the `IRR.RFACTOR.MFADEF.AZFTOTP1` profile in the RACF FACILITY Class:

- Create Attributes: ALTER
- Modify Attributes: UPDATE
- View Attributes: READ
- Delete Attributes: ALTER

Having selected the T option, the AZFTOTP1 Plug-in Attributes panel appears, as shown in Figure 4-2.

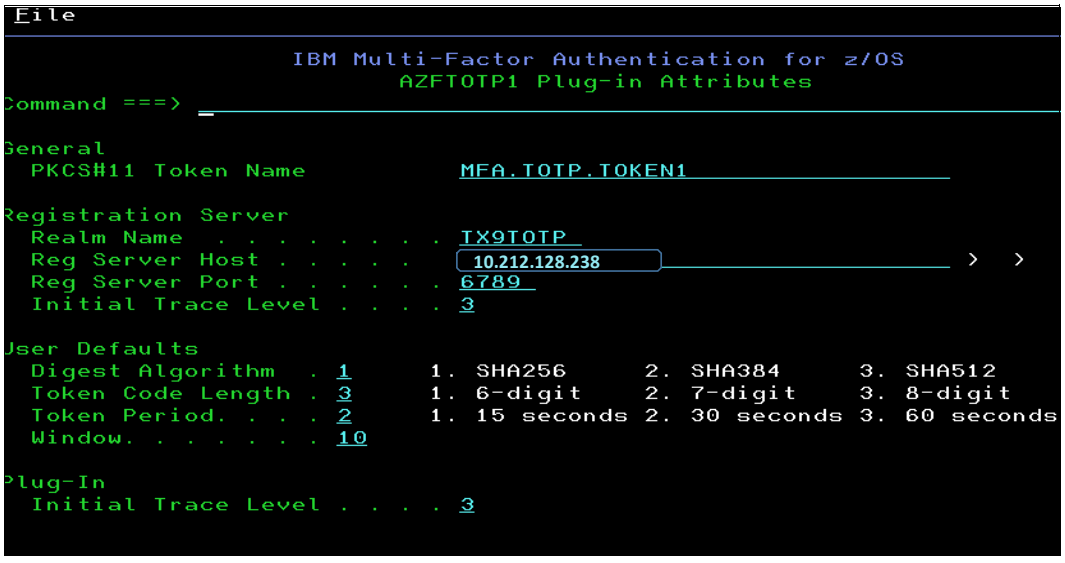

Figure 4-2 Configuring the AZFTOTP1 Plug-in attributes

4. Enter the following information:
   - PKCS#11 Token Name: The name of the PKCS#11 token to be used for AZFTOTP1 cryptographic operations. You created this token in 3.4, “Configuring a PKCS#11 token” on page 43.
   - Realm Name: The realm name for your AZFTOTP1 registration server. This name is an arbitrary name of your choosing.

– Registration Server Host: The host name (or IP address) of your AZFTOTP1 registration server. The host name must be sufficiently qualified for the iOS device users to resolve the host name.

  The host name or IP address should match the Subject Alternative Name extension of the certificate that is used for the registration server. Otherwise, the iOS browser requires the user to accept the certificate.

– Registration Server Port: The port number on which the registration server is listening. The port *must* match the one configured for AT-TLS.

– Initial Trace Level: The value to be used for tracing events within the AZFTOTP1 registration server. Valid values are 0 - 3, where the higher number increases the level of verbosity.

– Digest Algorithm: The default digest algorithm. IBM TouchToken uses the digest algorithm, shared secret key, and current time to generate the TOTP value. Possible values are SHA256, SHA384, and SHA512 (default is SHA256).

– Token Code Length: The number of digits in the generated token. Possible values are 6, 7, and 8 (default is 8).

– Token Period: The time (in seconds) between changes in value for the token. This value determines how long a one-time password is active before the next one-time password generates. Possible values are 15, 30, and 60 (default is 30).

– Window: The skew intervals of the algorithm. The skew intervals consider any possible synchronization delay between the server and the client that generates the one-time password. For example, a skew interval of 2 means a one-time password in up to two intervals in the past (or two in the future) are also valid. If it is interval 563 and intervals are 30 seconds, one-time passwords for intervals 561 - 565 are computed and checked against within a range of 2.5 minutes. The default is 1, and the maximum is 10.

Save your changes after you complete your choices.

The panel does not show how to save changes. No PF Keys are assigned to that task. The PF Key settings are shown in Example 4-2.

Example 4-2 Panel PF Key settings

```
F1=Help    F3=Exit    F4=Expand  F7=Up      F8=Down    F10=Left   F11=Right
F12=Cancel
```

To save your changes, press PF3. Upon exit, you are prompted with the following options:

– ENTER: Save the data
– CANCEL: Return to editing
– EXIT: Discard changes

5. After you configure IBM TouchToken, start the registration server AZF#IN01 by using an MVS **`Start MVS`** command. The **`Start`** command and an extract of the startup messages are shown in Example 4-3.

Example 4-3 Starting the registration service

```
S AZF#IN01
$HASP100 AZF#IN01 ON STCINRDR
IEF695I START AZF#IN01 WITH JOBNAME AZF#IN01 IS ASSIGNED TO USER AZFSTC
 , GROUP AZFGRP
$HASP373 AZF#IN01 STARTED
IEF403I AZF#IN01 - STARTED - TIME=13.01.34
...
AZFTOTPWEB:AZF500I IBM TouchToken Registration Server (level AZF-176, compiled
```

```
AZFTOTPWEB:AZF5009I AZFTOTP1 settings follow:
AZFTOTPWEB: Authenticator settings:
AZFTOTPWEB:  Initial trace level: 3
AZFTOTPWEB:  PKCS#11 token name:  MFA.TOTP.TOKEN1
AZFTOTPWEB:  Default ALG:         SHA256
AZFTOTPWEB:  Default NUMDIGITS:   8
AZFTOTPWEB:  Default PERIOD:      30
AZFTOTPWEB:  Default WINDOW:      5
AZFTOTPWEB: Registration server settings:
AZFTOTPWEB:  Initial trace level: 3
AZFTOTPWEB:  Realm name:          TX9TOTP
AZFTOTPWEB:  Host:  10.212.128.238
AZFTOTPWEB:  Port:                6789
AZFTOTPWEB:AZF5014I Will declare the following Realm name to clients: TX9TOTP
AZFTOTPWEB:AZF5015I Registration server using trace level: 3
AZFTOTPWEB:iezcom storage=0xfae10c
AZFTOTPWEB:MAIN loop EXTRACT, iezcom=0xfae10c
C5FC0 00000000                                          |......{....|
AZFTOTPWEB:AZF5050I Console listener task starting up
AZFTOTPWEB:AZF5002I Server base init success (port 6789)
```

The base TouchToken support is now completed for z/OS.

4.3 Installing the shared secret on the Apple iOS device

The following prerequisites must be satisfied to install the shared secret on the Apple iOS device:

- IBM TouchToken for iOS requires Touch ID. To use Touch ID, the following tasks must be completed:
  - Enable a passcode and enroll one or more fingerprints.
  - Ensure that your iOS device uses a complex alphanumeric passcode. If you do not have a complex alphanumeric passcode set on your iOS device, select on your phone **Settings** → **Touch ID and Passcode** → **Turn Passcode On (or Change Passcode)** → **Passcode Options** → **Custom Alphanumeric Code** to set one.
- IBM TouchToken is secured with TLS. Depending on the TLS configuration of the IBM TouchToken registration server, your security administrator might instruct you to download and install another Root CA certificate to a Configuration Profile in the iOS device. You can then optionally view this profile from the iOS device by selecting **Settings** → **General** → **Profile page**.

> **Note:** Be sure to download and install another Root CA certificate only with explicit guidance from your security administrator.

The procedure that is described in *IBM Multi-Factor Authentication for z/OS Installation and Customization,* SC27-8447, assumes that you are using a public CA. It is recommended that your IBM TouchToken registration server use a CA that is issued by a well-known CA. If you are not using a CA that is trusted by default by Apple iOS, you must ensure that all IBM TouchToken for iOS devices include an installed Configuration Profile that allows them to establish TLS connections with the registration server.

Note: If your registration server certificate was not issued by a well-known CA, do not instruct users to visit the registration server start page until they include an installed Configuration Profile that allows them to establish TLS connections with the registration server.

If users accept the registration server certificate in Mobile Safari as an SSL exception, the IBM TouchToken for iOS application still cannot trust the CA that issued the certificate. Although users can view the enrollment launch URL, they *cannot* complete the enrollment process.

We used a controlled laboratory to build the environment that is described in this IBM Redpaper publication. Because we used our own CA certificates to sign our MFA registration server certificate, a new profile must be configured in the iOS device to create the shared secret.

Although Apple iPads and iPhones can communicate with back-end servers securely in many ways, IT must configure the devices to accept valid CA certificates. There are several methods that can be used to add the certificates to iOS devices.

Every secure connection to the network starts with authentication to verify the server's identity. Most iPads and iPhones are configured to accept valid certificates that are issued by a trusted CA so the devices can determine which network servers are legitimate. IT Security (or appointed deputy) must follow a few steps to configure CA certificates for iPads and iPhones.

After a user is issued a passport (or a server is issued a certificate), these credentials can be presented with a signature as proof of identity. This kind of CA certificate validation occurs when a user browses a Secure Sockets Layer (SSL)-protected website. When validating the web server's certificate, the browser also checks the issuing CA's signature. This check often passes because public-facing websites tend to have CA certificates from one of the trusted root CAs that are configured by default into every operating system.

For more information about certificates, see *Managing Digital Certificates Across the Enterprise,* SG24-8336, which is available at this website:

`http://www.redbooks.ibm.com/abstracts/sg248336.html`

Applications that are running on iPads and iPhones can authenticate corporate servers by using privately issued certificates that provide instructions to trust them.

One high-risk option is to enable users to accept unknown CA certificates. However, by making such exceptions, users can be fooled self-signed certificates and those certificates that are issued by untrustworthy CAs. Allowing such exceptions can expose devices indefinitely to attacks.

A better option is for IT Security to explicitly add a trusted CA certificate to employee devices and configure applications to recognize and trust servers that prove their identity by using your company's CA certificates. In this way, IT Security can permit secure connections to trustworthy servers only.

All Apple iPads and iPhones support PKCS1-formatted X.509 certificates, which are stored in files that include the extensions `.crt`, `.cer`, `.pem`, or `.der`. You can use these certificates to identify CAs, servers, or individual users and devices.

You can add CA certificates and deploy configuration profiles that are used during enterprise web, email, VPN, or wireless LAN (WLAN) server authentication in the following ways:

- By using the web
- By using email
- By connecting an iOS device to a computer that is running the iPhone Configuration Utility
- By pushing profiles to an iOS device workgroup by using Apple Configuration (which is ideal for small organizations with fewer than 30 devices)
- Over the air by using an MDM tool (most businesses should consider MDM for fully automated, user-transparent iOS configurations)

Note: Use caution when email is used to send your trusted CA certificates to employees. When this method is used, instruct your users to make a one-time exception and to never install any other CA certificates, even if they appear to be from the IT department.

Consider sending your CA certificate as an attachment to an email that you send to your Apple ID or any email address that is a registered mail account on your iOS device.

We used the following method:

1. Remember that you copied your CA certificate as described in 3.4.1, “Creating a CA keyring and certificate” on page 43 by using the following command:

   ```
   OPUT 'PRICHAR.MFA.CACERT.DER' '/u/prichar/mfacacert.der' binary
   ```

2. Transfer this `mfacacert.der` file in binary mode to your workstation by using FTP.
3. Open your email application and attach this `mfacacert.der` file to an email.
4. Send this email to your device-registered email account.
5. In the iOS device email application, open the email and click the attachment. An Install Profile window opens. In this window, a message warns the user that the CA certificate that is about to be installed is not trusted. If the user selects the Install option, they are warned that the authenticity of the subject cannot be verified and that installing the profile adds it to the list of trusted certificates on that iPad or iPhone.

   The name of the profile is `RACF CA for MFA server` and it is from the subject common name “CN” that we used in the RACDCERT command as shown in Example 4-4.

   Example 4-4 Previously defined certificate command

   ```
   RACDCERT CERTAUTH GENCERT SUBJECT(CN('RACF CA for MFA server')  +
      OU('MFA TX9') O('IBM') ) +
      NOTAFTER(DATE(2025/12/31)) WITHLABEL('MFA TX9 RACF CA')
   ```

The window that is shown in Figure 4-3 opens on the iOS device. Select the **Install** option.

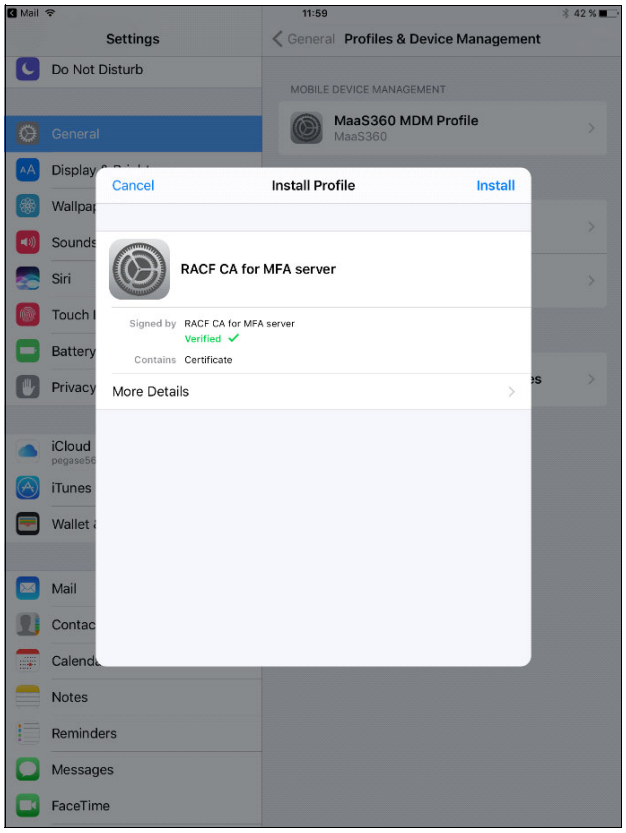

Figure 4-3 Image of iOS display to install certificate

6. After you install the certificate, select **Settings** → **General** → **Profiles & Device Management**, as shown in Figure 4-4 on page 71.

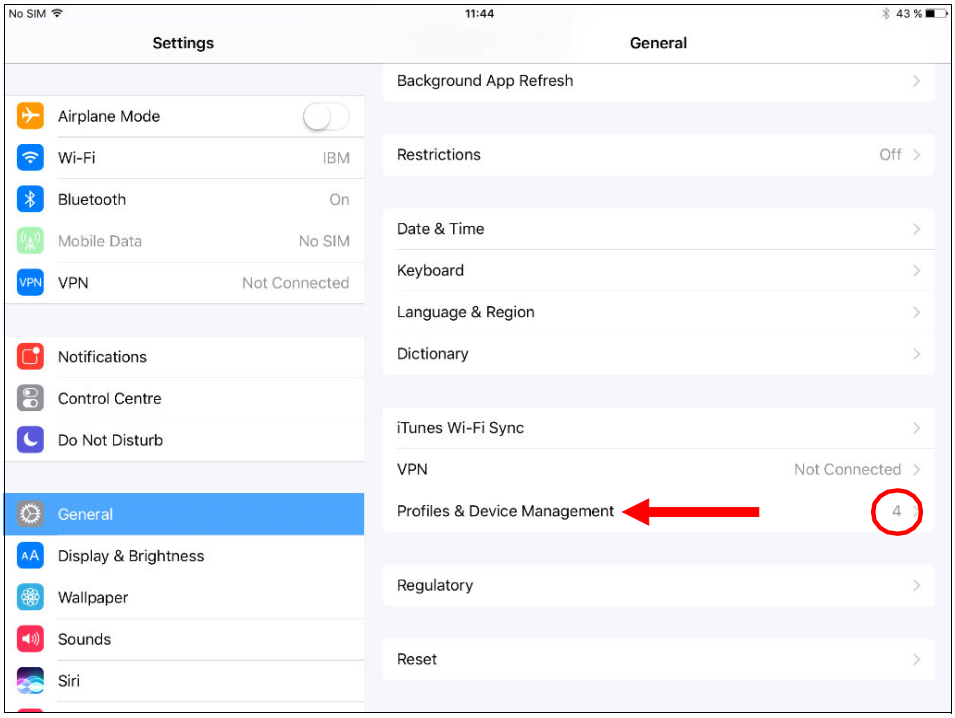

Figure 4-4 Profiles & Device Management on iOS device

Four entries are listed in the Profiles and Device Management window. Select the arrow to view these entries (see Figure 4-5). We overwrote the IP address in our example.

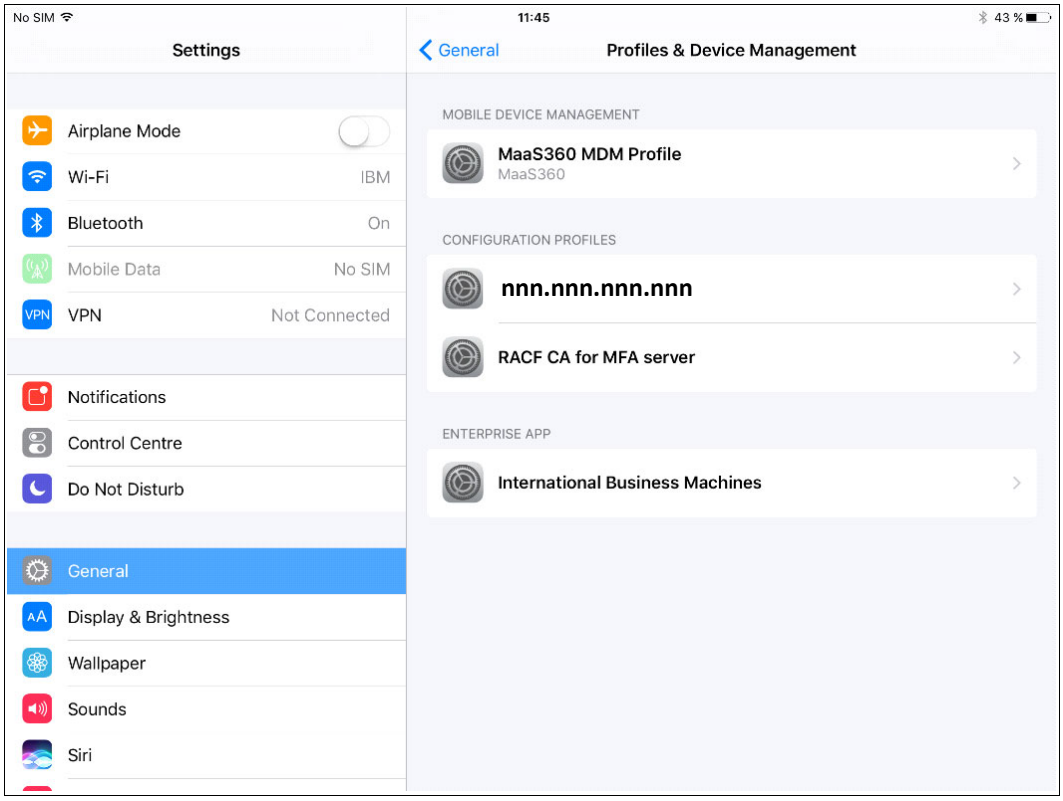

Figure 4-5 Profiles & Device Management entries

If you select the Profiles view, the information that is shown in Figure 4-6 is displayed.

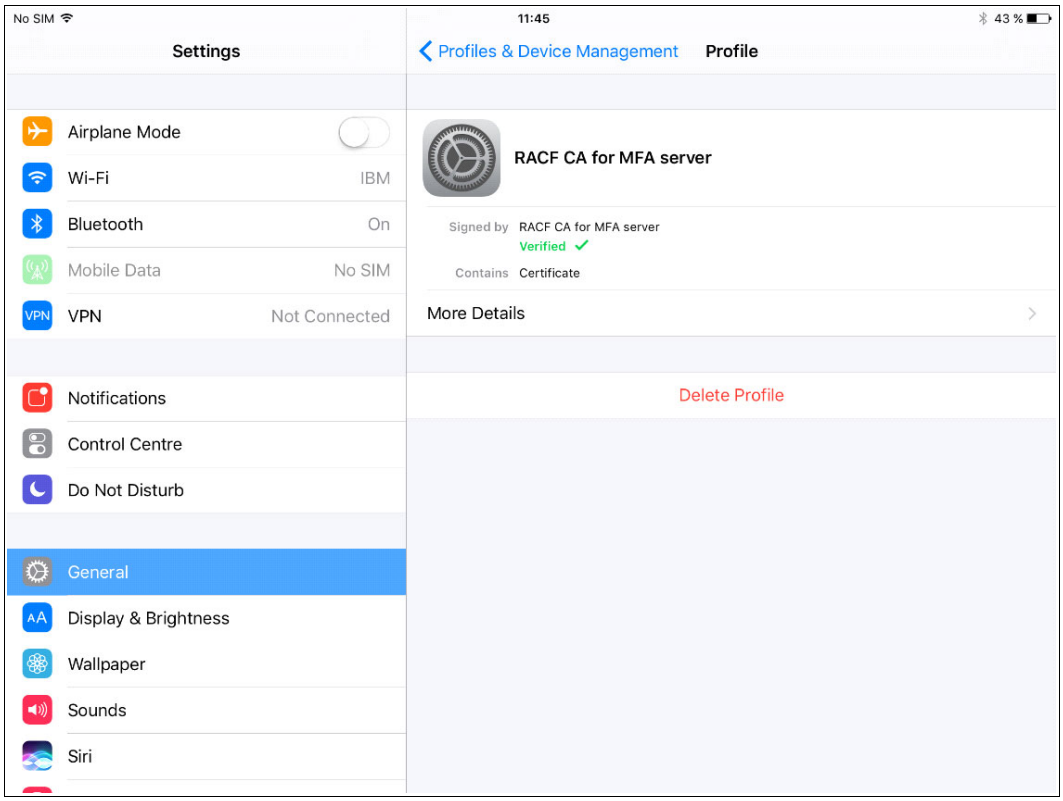

Figure 4-6 Profile display

You can see that the RACF CA for the MFA server was verified.

If you select the **More Details** option, the display that is shown in Figure 4-7 displays.

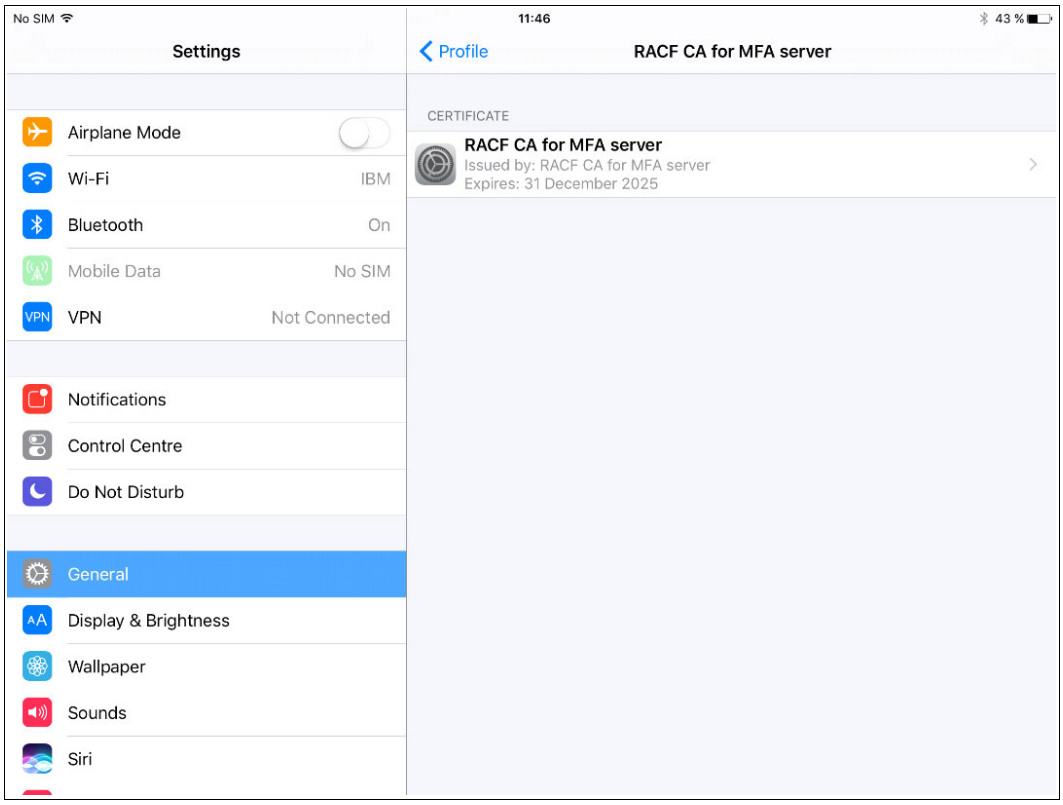

Figure 4-7 RACF CA for MFA server profile

Select the arrow to see profile details, as shown in Figure 4-8.

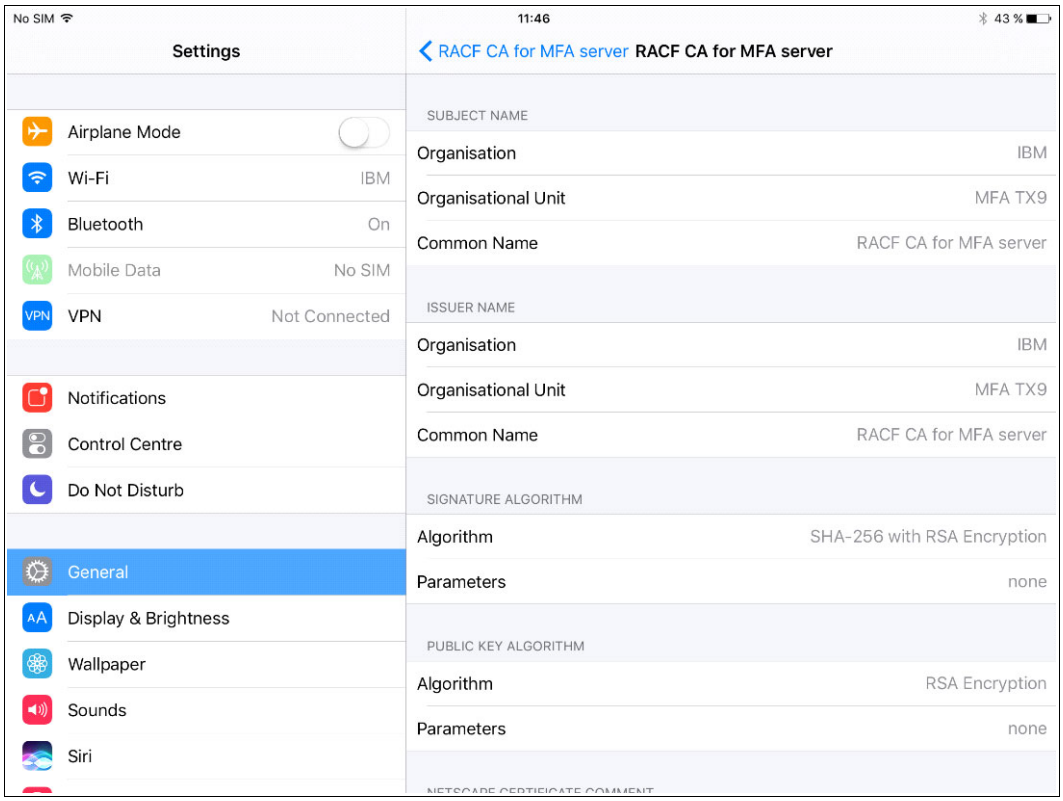

Figure 4-8 Profile details

The details that are on the iOS device match the **RACDCERT** command that was used generated the certificate. The iOS configuration work in preparation for IBM TouchToken is now completed.

4.3.1 Downloading and installing the IBM TouchToken for iOS

The last phase is to download and install the IBM TouchToken for iOS application. For more information, see the Apple store, as shown in Figure 4-9.

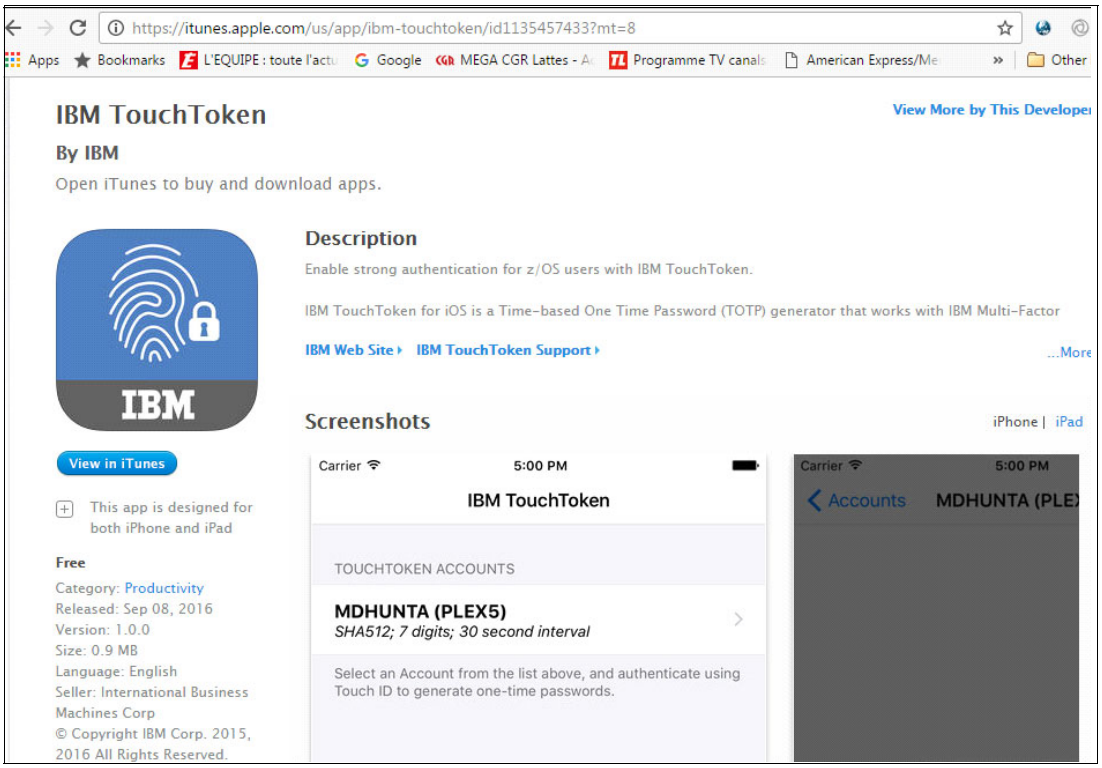

Figure 4-9 IBM TouchToken for iOS

The download is available from the following iTunes website, as shown in Figure 4-10:

`https://itunes.apple.com/us/app/ibm-touchtoken/id1135457433?mt=8`

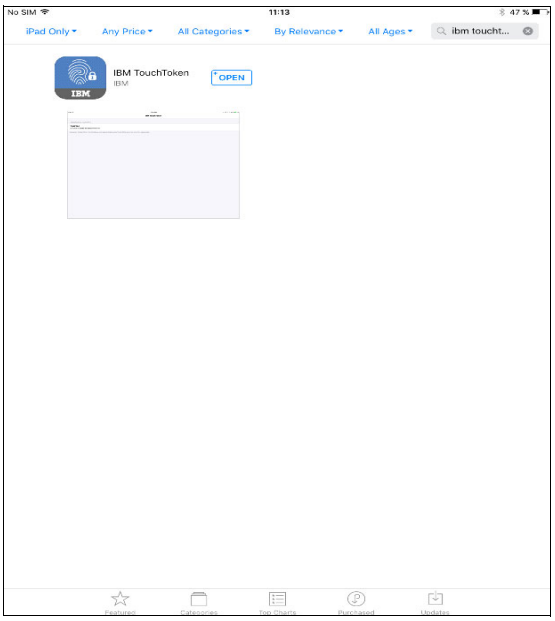

Figure 4-10 Downloaded application

After the application is downloaded and installed, a blue icon application that features a padlock is displayed on your device, as shown in Figure 4-11.

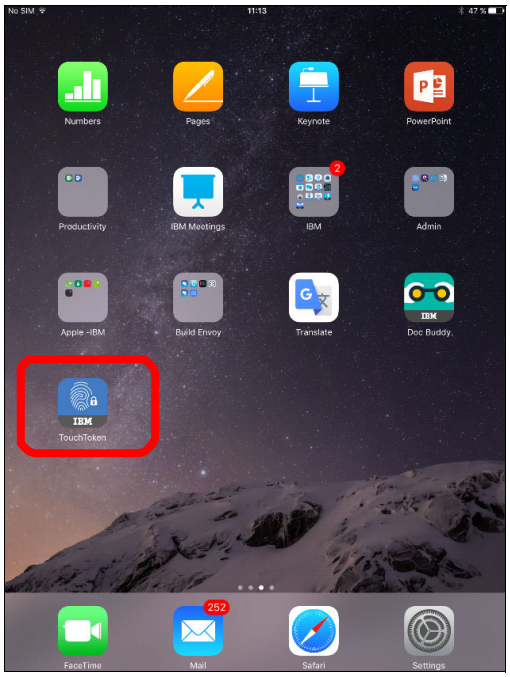

Figure 4-11 IBM TouchToken icon

5

User account administration and logon

In this chapter, we describe how to configure the IBM TouchToken for user accounts. Then, we describe how the configuration options can be used to logon to z/OS applications.

This chapter includes the following topics:

5.1 Configuring user accounts for IBM TouchToken

Before a user can activate the IBM TouchToken for iOS application, you must configure the IBM TouchToken profile for the user. This configuration allows IBM TouchToken to register the IBM TouchToken for iOS application. To complete this configuration process, use the **ALTUSER** (**ALU**) command.

When a User ID is not a registered MFA account, the user's MFA segment in RACF is not defined or is inactive, as shown in Example 5-1.

Example 5-1 Listing the user's attributes

```
LISTUSER DRICHAR  MFA
USER=DRICHAR  NAME=DOMINIQUE RICHARD     OWNER=MVS       CREATED=13.064
 DEFAULT-GROUP=OMVSGRP  PASSDATE=16.281 PASS-INTERVAL= 60 PHRASEDATE=N/A
 ATTRIBUTES=SPECIAL OPERATIONS
 REVOKE DATE=NONE   RESUME DATE=NONE
 LAST-ACCESS=16.293/07:44:55
 CLASS AUTHORIZATIONS=NONE
 NO-INSTALLATION-DATA
 NO-MODEL-NAME
 LOGON ALLOWED   (DAYS)          (TIME)
 -----------------------------------------
 ANYDAY                          ANYTIME
  GROUP=OMVSGRP   AUTH=CREATE   CONNECT-OWNER=MVS       CONNECT-DATE=13.064
    CONNECTS=    55  UACC=NONE     LAST-CONNECT=16.292/16:23:01
    CONNECT ATTRIBUTES=NONE
    REVOKE DATE=NONE   RESUME DATE=NONE
  GROUP=MVS       AUTH=CREATE   CONNECT-OWNER=PRICHAR   CONNECT-DATE=13.064
    CONNECTS=    00  UACC=NONE     LAST-CONNECT=UNKNOWN
    CONNECT ATTRIBUTES=NONE
    REVOKE DATE=NONE   RESUME DATE=NONE
...
...
THERE IS NO MULTIFACTOR AUTHENTICATION DATA.
***
```

Complete the following steps to set up the user for MFA:

1. Enter the command that is shown in Example 5-2 to set the IBM TouchToken registration state for the user to OPEN.

   Example 5-2 Setting the IBM TouchToken registration state to open

   ```
   ALU [Login ID] MFA(FACTOR(AZFTOTP1)
   TAGS(REGSTATE:OPEN))
   ```

   Note: The **O**PEN is case-sensitive.

2. Enter the **LISTUSER** command to display MFA information for a user profile. The command and response are shown in Example 5-3.

Example 5-3 Displaying the MFA information

```
LISTUSER DRICHAR  MFA
USER=DRICHAR  NAME=DOMINIQUE RICHARD     OWNER=MVS       CREATED=13.064
 DEFAULT-GROUP=OMVSGRP  PASSDATE=16.281 PASS-INTERVAL= 60 PHRASEDATE=N/A
 ATTRIBUTES=SPECIAL OPERATIONS
 REVOKE DATE=NONE   RESUME DATE=NONE
 LAST-ACCESS=16.293/07:44:55
 CLASS AUTHORIZATIONS=NONE
 NO-INSTALLATION-DATA
 NO-MODEL-NAME
 LOGON ALLOWED   (DAYS)          (TIME)
 -----------------------------------------
 ANYDAY                          ANYTIME
  GROUP=OMVSGRP   AUTH=CREATE   CONNECT-OWNER=MVS       CONNECT-DATE=13.064
    CONNECTS=    55  UACC=NONE     LAST-CONNECT=16.292/16:23:01
    CONNECT ATTRIBUTES=NONE
    REVOKE DATE=NONE   RESUME DATE=NONE
  GROUP=MVS       AUTH=CREATE   CONNECT-OWNER=PRICHAR   CONNECT-DATE=13.064
...
...
MULTIFACTOR AUTHENTICATION INFORMATION:
---------------------------------------
PASSWORD FALLBACK IS NOT ALLOWED
  FACTOR = AZFTOTP1
    STATUS = INACTIVE
    FACTOR TAGS =
      REGSTATE:OPEN
***
```

The MFA information is displayed. The AZFTOTP1 authentication factor is INACTIVE, so the user must provide their RACF credentials (User ID and password) to authenticate to any z/OS application.

3. Ensure the user's devices are set up for IBM TouchToken for IBM:
   - Check that the user's Apple iOS device features network connectivity to the IBM TouchToken registration server.
   - Check that the users installed the IBM TouchToken for iOS application on their iOS device.
   - Check that the users can open the IBM TouchToken registration server start page by using Mobile Safari on their iOS device and log in with their z/OS RACF username and password.

     The website `https://hostname:6789/AZFTOTP1/start` provides more information about IBM TouchToken for the user and features a link that starts the IBM TouchToken for iOS application on the user's device, as shown in Figure 5-1 on page 80.

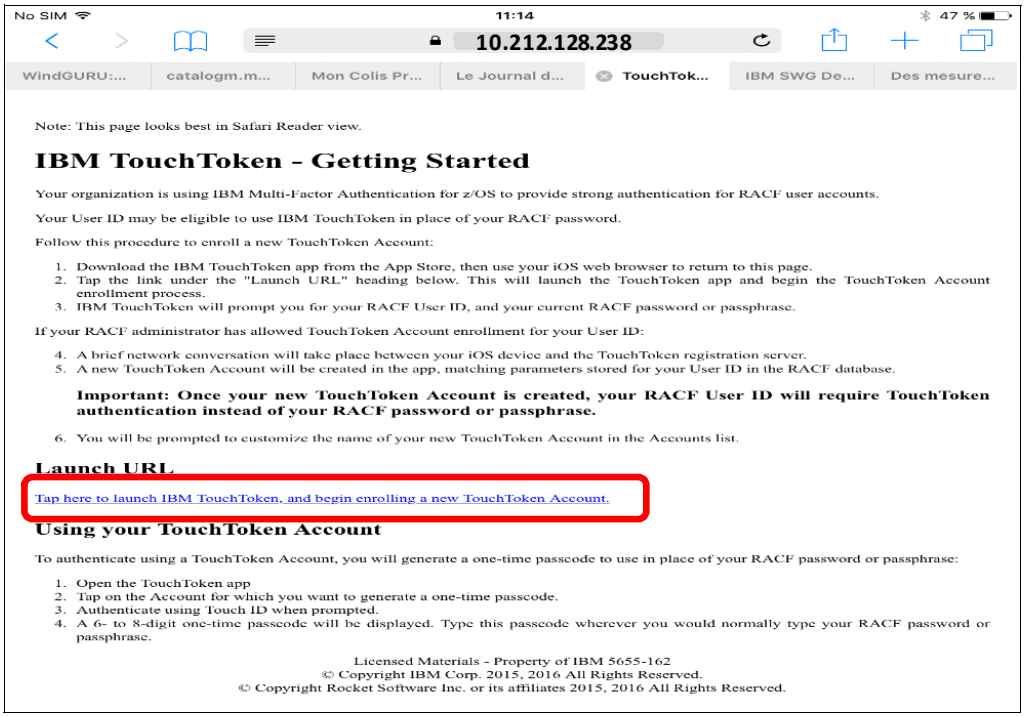

Figure 5-1 IBM TouchToken Getting Started page

4. Tap the link. A confirmation message appears. Confirm that you want to open the application. The New Account Enrollment page that is shown in Figure 5-2 opens.

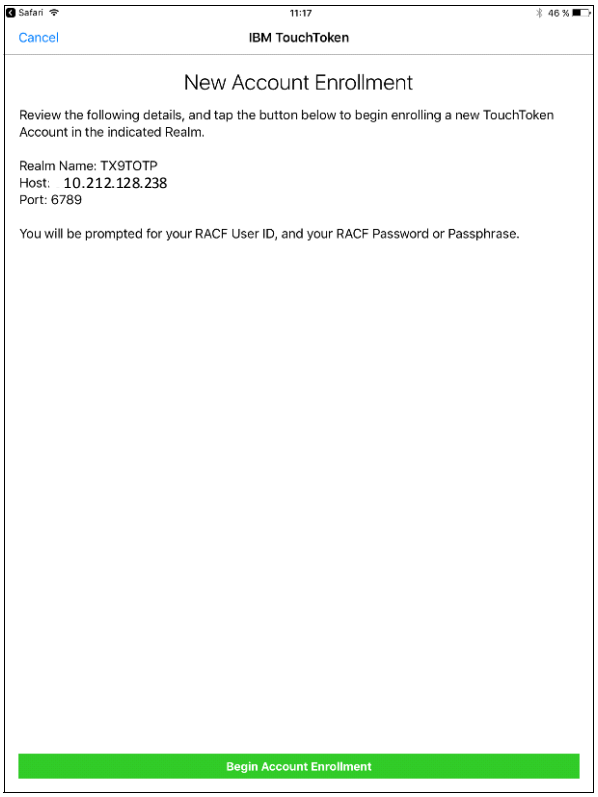

Figure 5-2 New Account Enrollment page on iOS device

Follow the on-screen instructions and tap the **Begin Account Enrollment** option at the bottom of the window.

5. The window that is shown in Figure 5-3 opens. Enter your RACF User ID and password.

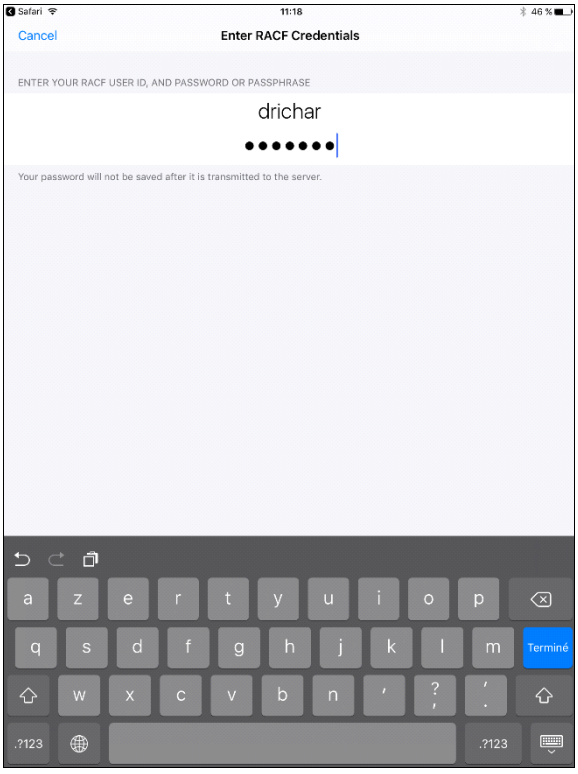

Figure 5-3 Entering RACF credentials

When prompted, enter your Apple TouchID fingerprint.

6. Set an account alias. This setting is useful because it is an opportunity to remove any system-specific information, as shown in Figure 5-4 on page 82.

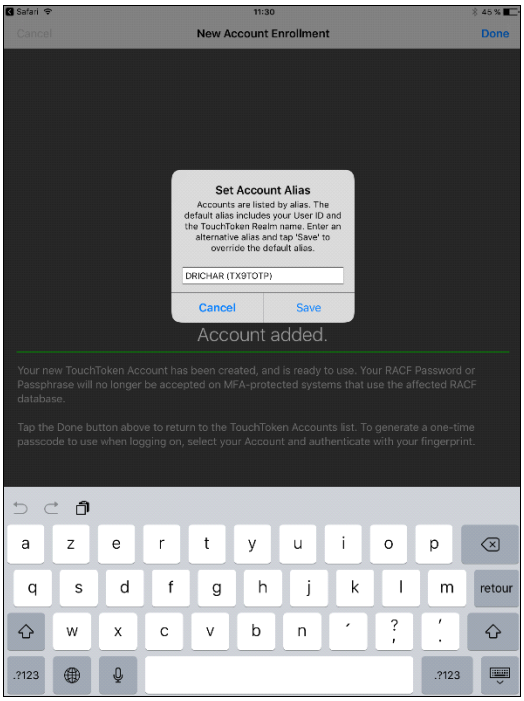

Figure 5-4 Setting an Account Alias

Enter the alias information and tap **Save**. The window that is shown in Figure 5-5 opens.

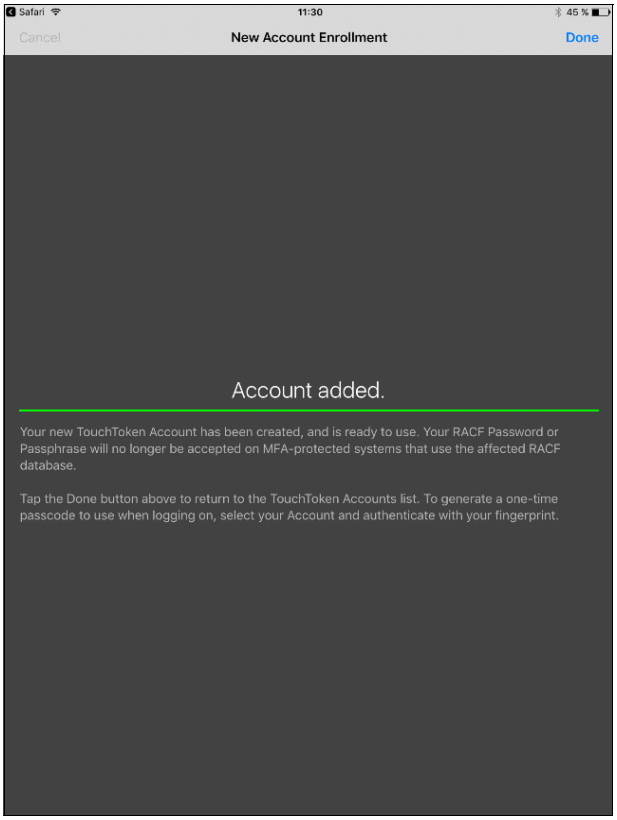

Figure 5-5 Account added confirmation

Note: Now that the user account is added, the user's password or passphrase is no longer be accepted on MFA active systems that are attached to the active RACF database that contains the appropriate MFA and User profile settings.

7. Tap **Done** and the new window shows the user's TouchToken accounts, as shown in Figure 5-6.

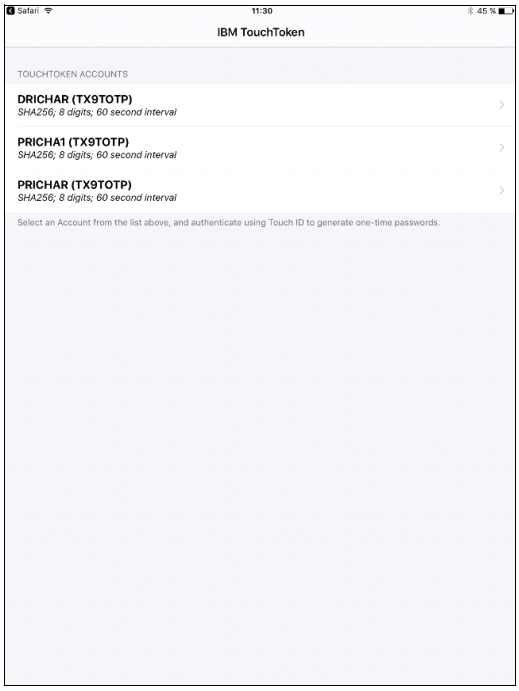

Figure 5-6 User's TouchToken accounts

Note: After the IBM TouchToken account is set up on the Apple device, the RACF REGSTATE in the user's MFA segment changes to PROVISIONED.

To generate a TOTP for logging on, the user selects the account that they want to use and authenticate this part by using their fingerprint.

5.2 Verifying the user account is ready to use TOTP token codes

Before generating the TOTP passcode, verify that the user account is properly configured by completing the following steps:

1. Issue two **LISTUSER** commands to check that the user has the MFA segment defined. Example 5-4 shows the command and response. The first command lists the user. A message indicates that the data exists and how to display it.

Example 5-4 Using LISTUSER commands to find user information

```
LISTUSER DRICHAR
USER=DRICHAR  NAME=DOMINIQUE RICHARD     OWNER=MVS       CREATED=13.064
  DEFAULT-GROUP=OMVSGRP  PASSDATE=16.281 PASS-INTERVAL= 60 PHRASEDATE=N/A
  ATTRIBUTES=SPECIAL OPERATIONS
  REVOKE DATE=NONE   RESUME DATE=NONE
  LAST-ACCESS=16.293/10:14:20
  CLASS AUTHORIZATIONS=NONE
  NO-INSTALLATION-DATA
...
...
MULTIFACTOR AUTHENTICATION DATA EXISTS. USE THE MFA KEYWORD TO DISPLAY IT.

LISTUSER DRICHAR  MFA
MULTIFACTOR AUTHENTICATION INFORMATION:
 -------------------------------------
   PASSWORD FALLBACK IS NOT ALLOWED
   FACTOR = AZFTOTP1
     STATUS = ACTIVE
     FACTOR TAGS =
       REGSTATE:PROVISIONED
       KEYLABEL:AZF.DRICHAR.D171603F29653359
       ALG:SHA256
       CVALUE:24611964
       NUMDIGITS:8
       PERIOD:60
       WINDOW:10
```

2. Record the KEYLABEL information in the command response.
3. Browse to the ICSF Token Management application. Complete the following steps:
   a. In your mainframe TSO session, go to the ICSF ISPF application, as shown in Example 5-5.

Example 5-5 Select the ICSF Facility

```
OPTION ===> ic                                          SCROLL ===> CSR

IS  ISMF     - Interactive Storage Management Facility
  P   PDF      - ISPF/Program Development Facility
  IP  IPCS     - Interactive Problem Control Facility
  AZ  MFA      - IBM Multi-Factor Authentication for z/OS
  DI  DITTO    - Data Interfile Transfer, Testing and Operations
  SD  SDSF     - System Display and Search Facility
  R   RACF     - Resource Access Control Facility
```

```
  HC  HCD      - Hardware Configuration Definition
  BMB BMR BLD  - BookManager Build (Create Online Documentation)
  BMR BMR READ - BookManager Read (Read Online Documentation)
  BMI BMR INDX - BookManager Read (Create Bookshelf Index)
  SM  SMP/E    - SMP/E Dialogs
  IC  ICSF     - Integrated Cryptographic Service Facility
  OS  SUPPORT  - OS/390 ISPF System Support Options
  OU  USER     - OS/390 ISPF User Options
```

b. On the ICSF menu, select **option 5** to access the ICSF Utilities, as shown in Example 5-6.

Example 5-6 Accessing the ICSF Utilities

```
HCR77B0  -------------- Integrated Cryptographic Service Facility --
 OPTION ===> 5
 Enter the number of the wanted option.

   1  COPROCESSOR MGMT -  Management of Cryptographic Coprocessors
   2  KDS MANAGEMENT   -  Master key set or change, KDS Processing
   3  OPSTAT           -  Installation options
   4  ADMINCNTL        -  Administrative Control Functions
   5  UTILITY          -  ICSF Utilities
   6  PPINIT           -  Pass Phrase Master Key/KDS Initialization
   7  TKE              -  TKE PKA Direct Key Load
   8  KGUP             -  Key Generator Utility processes
   9  UDX MGMT         -  Management of User Defined Extensions
```

c. The ICSF Utilities menu is displayed. Select **option 7** to access the PKCS11 token management, as shown in Example 5-7.

Example 5-7 Accessing PKCS11 Token Management

```
------------------------------ ICSF - Utilities ----
 OPTION ===> 7

 Enter the number of the wanted option.

   1  ENCODE       -  Encode data
   2  DECODE       -  Decode data
   3  RANDOM       -  Generate a random number
   7  PKCS11 TOKEN -  Management of PKCS11 tokens
```

d. The ICSF Token Management menu is displayed. Select **option 4** to list the tokens, as shown in Example 5-8.

Example 5-8 Option to list tokens

```
---------------------- ICSF Token Management - Main Menu --
OPTION ===> 4
Enter the number of the wanted option.
  1  Create a new token
  2  Delete an existing token
  3  Manage an existing token
  4  List existing tokens

  Full or partial token name:
```

e. Select the option to manage the `MFA.TOTP.TOKEN1` by entering an M to next to it, as shown in Example 5-9.

Example 5-9 Selecting to manage the MFA.TOTP.TOKEN1

```
COMMAND ===>

   Select a token to manage(M) or delete(D) then press ENTER

   Press END to return to the previous menu.

 m MFA.TOTP.TOKEN1
 _ PKISRVD.PKITOKEN
 _ SYSTOK-SESSION-ONLY
 _ TAMPER.RESISTANT.SMF.TOKEN.NAME1
```

f. The KEYLABEL that was recorded from the LISTUSER response in step 2 is displayed and matches the recorded information, as shown in Example 5-10.

Example 5-10 Checking the KEYLABEL

```
------------------- ICSF Token Management - Token Details -- Row 2 to 3 of 6
 COMMAND ===>                                                  SCROLL===>CSR

 Token name: MFA.TOTP.TOKEN1
 Manufacturer: RACF HRF77A0
 Model: HCR77B0
 Serial Number: 0
 Number of objects: 6

 Select objects to process then press ENTER

 Press END to return to the previous menu.

 ------------------------------------------------------------------------------
 _ Object 0000000DT SECRET KEY    PRIVATE: TRUE      MODIFIABLE: TRUE
                                 EXTRACTABLE: TRUE   SENSITIVE: FALSE
    LABEL:          AZF.DRICHAR.D171603F29653359
    ID:             <Not-specified>
    KEY TYPE:       GENERIC SECRET
    VALUE LEN:      64
    USAGE FLAGS:
Enc(T),Sign(T),Wrap(T),Derive(T),Dec(T),Verify(T),Unwrap(T)
```

This shared secret key is used to derive the TOTP token code.

IBM TouchToken uses the following elements to generate the TOTP value:

- Digest algorithm (possible values are SHA256, SHA384, and SHA512)
- Shared secret key
- Current time

5.3 Generating a one-time passcode

IBM TouchToken combines Touch ID fingerprint biometric technology with a hashed, timed one-time password (OTP) for secure multi-factor authentication. You can consider the IBM TouchToken OTP to be a token code. For more information, see 1.2.2, “RSA authentication manager” on page 3.

The IBM TouchToken for iOS application on the Apple device uses a shared secret key and the current time to generate token code values. IBM TouchToken on the z/OS system then uses the same algorithm to validate user logons.

IBM TouchToken requires Apple Touch ID fingerprint technology. The token code that is generated by the IBM TouchToken for iOS application must fall within an approved sequence of token codes that are generated by the IBM TouchToken component on the z/OS server.

Token codes are regenerated at regular intervals. One-time token codes that are generated from the same secret key are identical if they are generated within a predetermined “token period” time value, which your security administrator can set at intervals of 15, 30, or 60.

To authenticate by using a TouchToken Account, a one-time passcode is generated to use in place of your RACF password or passphrase. Complete the following steps:

1. On the iOS device, open the TouchToken application. A list of IBM TouchToken accounts appears, as shown in Figure 5-7.

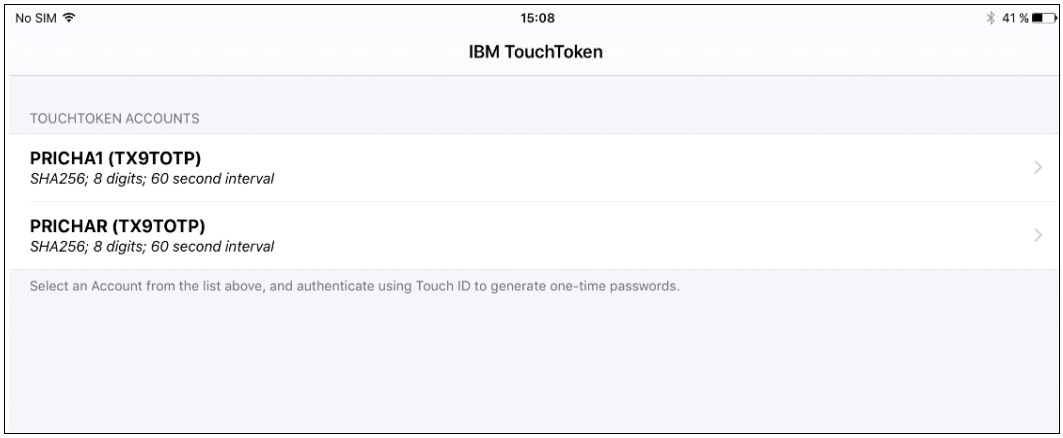

Figure 5-7 IBM TouchToken account list

2. Tap the account for which you want to generate a one-time passcode. The window that is shown in Figure 5-8 opens.

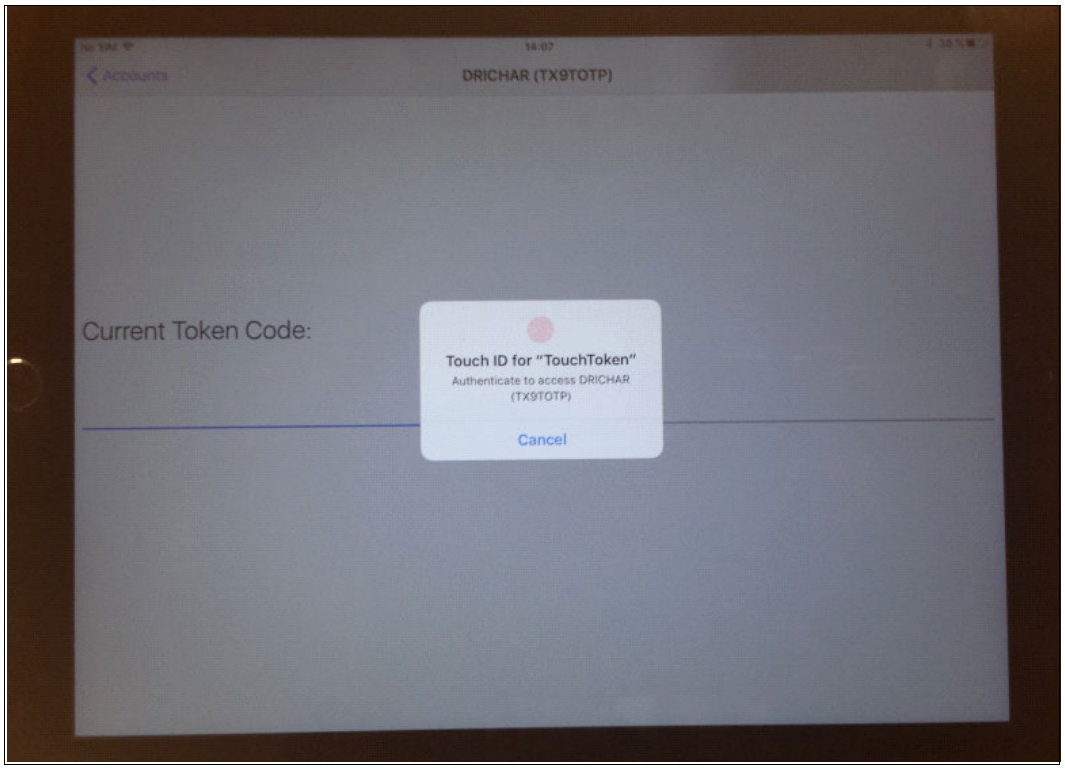

Figure 5-8 Authenticating by using fingerprint

3. You are prompted to authenticate by using Touch ID. Authenticate by using your fingerprint over the scanner on the device. If you do not touch the scanner properly or use the incorrect finger, the window that is shown in Figure 5-9 on page 89 opens.

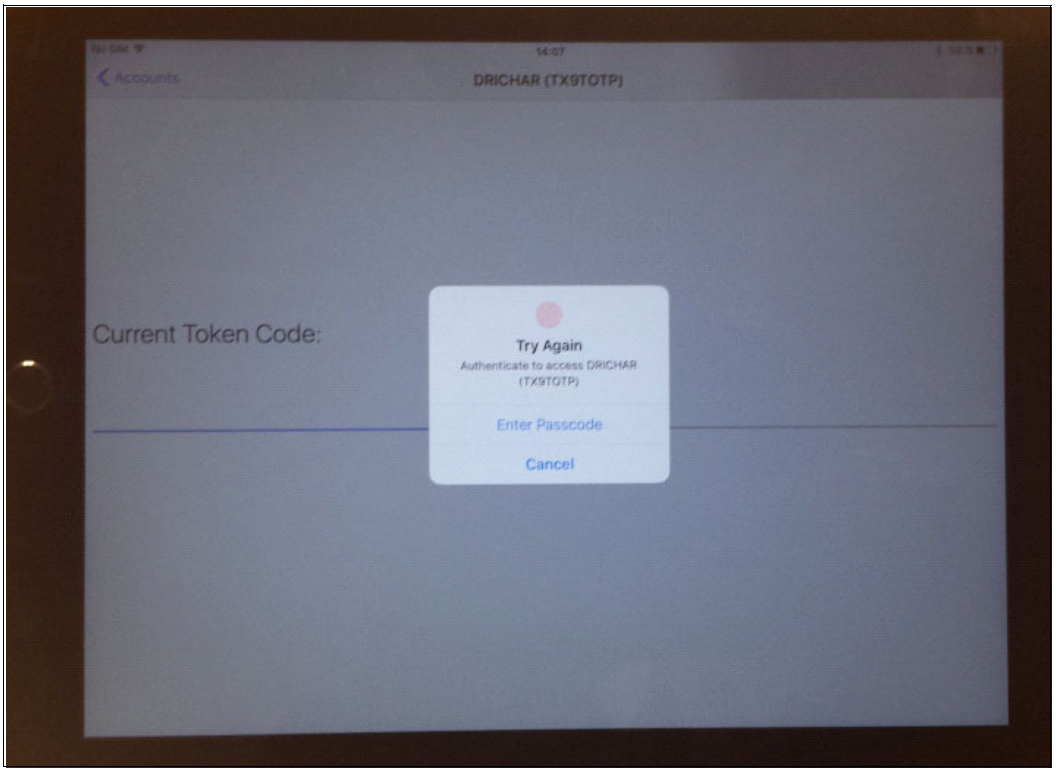

Figure 5-9 Try again to authenticate with your fingerprint

4. Retry authenticating your fingerprint. If successful, a 6 - 8 digit one-time passcode is displayed, as shown in Figure 5-10. Enter this passcode wherever you enter your RACF password or passphrase.

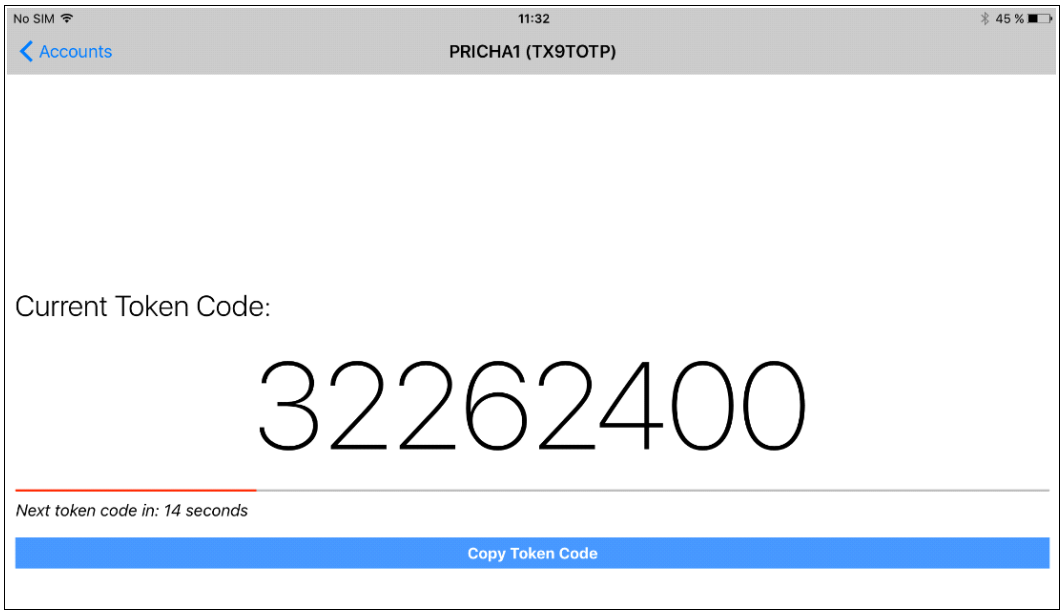

Figure 5-10 Eight-digit one-time passcode

Note: The generated token is valid for a limited time (our system default was set to 60 seconds). Because we waited for more than 46 seconds, a warning message appeared that read: “Next token code in: 14 seconds”. The displayed token expires after 60 seconds if it is not used and a new token is generated.

The user must now use this TOTP token code to log on to the z/OS system application.

Tip: It can take you longer than you expect to enter the password into your z/OS application. If the password expires, the authentication fails. To avoid this situation, wait for the new passcode to be generated and enter it into your z/OS application.

5.4 Logging on to z/OS application by using your TouchToken

In this section, we describe the following examples of logging on to a z/OS application:

- TSO
- z/OSMF
- MVS Console

Generally, enter your IBM MFA credentials if you are presented with a request for logon credentials. If you are presented with a request for logon credentials, try your IBM MFA credentials first even if your account is configured to use PassTickets because you might be required to first log on with IBM MFA.

In all three examples, you see what occurs if you log on by using your RACF credentials when your account is MFA active. Then, the same logon by using the generated passcode is attempted.

5.4.1 Logging on to TSO

The first attempt is by using RACF credentials only; the second attempt is by using the generated token code.

RACF credentials for TSO

If you attempt to logon to TSO and you enter your RACF credentials, the attempt fails. You see the messages and in the SYSLOG that are shown in Example 5-11.

Example 5-11 TSO logon failure messages by using RACF credentials

```
On the TSO logon screen you receive:
ICH70008I IBM MFA Message:
          AZF1104I TOTP PASSCODE REJECTED
 ***

In the SYSLOG, you see:
ICH408I USER(DRICHAR ) GROUP(OMVSGRP ) NAME(DOMINIQUE RICHARD   ) 419
  LOGON/JOB INITIATION - MULTIFACTOR AUTHENTICATION FAILURE
```

You can determine the relevant information that is issued by an authentication request. A trace is required to determine the relevant information that is issued by any particular authentication request that is made by an IBM MFA user.

To start the trace, issue the operator command that is shown in Example 5-12. AZF#IN00 is the MFA started task name. The response message also is shown.

Example 5-12 Starting the trace

```
F AZF#IN00,STC SET TRACE LEVEL 2
AZF2112I Console received modify command : STC SET TRACE LEVEL 2
```

Attempt to authenticate again. This time, the trace is active. Search the SYSLOG for entries that begin with MFAA, which is similar to text that is shown in Example 5-13.

Example 5-13 MFAA eyecatcher message in SYSLOG

```
MFAA Version=2, MFAA Length=264, Application=TESTAPP, STC UserID=TSTUSR
```

The IBM MFA started task's SYSPRINT contains lines that show the application name and user ID values that are supplied explicitly and implicitly by the issuer of the RACROUTE REQUEST=VERIFY call, which you can use to guide you in defining profiles.

In the MFA server (AZF#IN00) log, you see messages similar to the messages that are shown in Example 5-14.

Example 5-14 AZF#IN00 logon failed messages

```
AZFSTC:PC return code=0x8, reason code=0x5, abend code=0x0
AZFSTC:AZF2105I Authentication request (PC)
AZFSTC:MFAA Version=2, MFAA Length=264, Application=A6PTX0  , STC UserID=
AZFSTC:Password length=8, Passphrase length=0, New Password Length=0, New Passph
AZFSTC:authStateTransition mfaAuthTxn START (23AEE1E8)
AZFSTC:authStateTransition (23AEE1E8) TO txnResult=0x1 (pre-switch)
AZFSTC:AZF2202W In-band auth failed
AZFSTC:authStateTransition (23AEE1E8) TO txnResult=0x1 (post-switch)
AZFSTC:PC return code=0x8, reason code=0x0, abend code=0x0
```

In the messages that are shown in Example 5-14, you can identify the application to which you attempted to logon to (TSO application ID= A6PTX0, IBM VTAM® generic resource for my TSO, as defined in GNAME= in the TSOKEYxx PARMLIB member).

Using generated token code for TSO

Instead of using the RACF password or passphrase to log on to TSO, use the passcode you generated. The logon should be successful and you see messages in the AZF#IN00 server log that are similar to the messages that are shown in Example 5-15.

Example 5-15 Successful logon to TSO by using a generated token code

```
AZFSTC:AZF2105I Authentication request (PC)
AZFSTC:MFAA Version=2, MFAA Length=264, Application=A6PTX0  , STC UserID=
AZFSTC:Password length=8, Passphrase length=0, New Password Length=0, New Passph
AZFSTC:authStateTransition mfaAuthTxn START (23AEE1E8)
AZFSTC:authStateTransition (23AEE1E8) TO txnResult=0x1 (pre-switch)
AZFSTC:AZF2201I In-band auth success
AZFSTC:authStateTransition (23AEE1E8) TO txnResult=0x1 (post-switch)
AZFSTC:PC return code=0x0, reason code=0x0, abend code=0x0
```

5.4.2 Logging on to z/OSMF

In this section, the first attempt to log on to z/OSMF uses RACF credentials only; the second attempt uses the generated token code.

RACF credentials for z/OSMF

If you attempt to logon to z/OSMF by entering your RACF credentials, the attempt fails. You see messages that are similar to the messages that are shown in Figure 5-11.

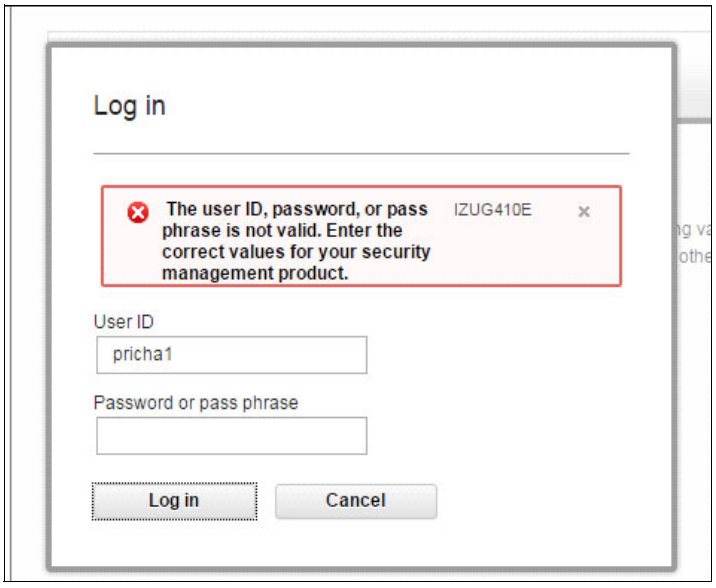

Figure 5-11 z/OSMF log on failure

Messages are generated in the SYSLOG and AZF#IN00 log that are similar to the messages that are shown in Example 5-16.

Example 5-16 Log on to z/OSMF failed messages

```
In the SYSLOG, you see:
ICH408I USER(DRICHAR ) GROUP(OMVSGRP ) NAME(DOMINIQUE RICHARD   ) 464
  LOGON/JOB INITIATION - MULTIFACTOR AUTHENTICATION FAILURE
IRR013I  VERIFICATION FAILED. INVALID PASSWORD GIVEN.

In the AZF#IN00 log, you see:
AZFSTC:PC return code=0x8, reason code=0x0, abend code=0x0
AZFSTC:AZF2105I Authentication request (PC)
AZFSTC:MFAA Version=2, MFAA Length=264, Application=IZUDFLT , STC UserID=IZUSVR
AZFSTC:Password length=8, Passphrase length=0, New Password Length=0, New Passph
AZFSTC:authStateTransition mfaAuthTxn START (23AEE1E8)
AZFSTC:authStateTransition (23AEE1E8) TO txnResult=0x1 (pre-switch)
AZFSTC:AZF2202W In-band auth failed
AZFSTC:authStateTransition (23AEE1E8) TO txnResult=0x1 (post-switch)
AZFSTC:PC return code=0x8, reason code=0x0, abend code=0x0
```

In the message that is shown in Example 5-16, the application is `Application=IZUDFLT, STC UserID=IZUSVR`.

Using generated token code for z/OSMF

Instead of using the RACF password or passphrase to log on to z/OSMF, use the passcode that you generated. The logon should be successful and you see messages in the AZF#IN00 server log that are similar to the messages that are shown in Example 5-17.

Example 5-17 Successful log on to z/OSMF by using a generated token code

```
AZFSTC:AZF2105I Authentication request (PC)
AZFSTC:MFAA Version=2, MFAA Length=264, Application=IZUDFLT , STC UserID=IZUSVR
AZFSTC:Password length=8, Passphrase length=0, New Password Length=0, New Passph
AZFSTC:authStateTransition mfaAuthTxn START (23AEE1E8)
AZFSTC:authStateTransition (23AEE1E8) TO txnResult=0x1 (pre-switch)
AZFSTC:AZF2201I In-band auth success
AZFSTC:authStateTransition (23AEE1E8) TO txnResult=0x1 (post-switch)
AZFSTC:PC return code=0x0, reason code=0x0, abend code=0x0
```

The z/OSMF Welcome window opens, as shown in Figure 5-12.

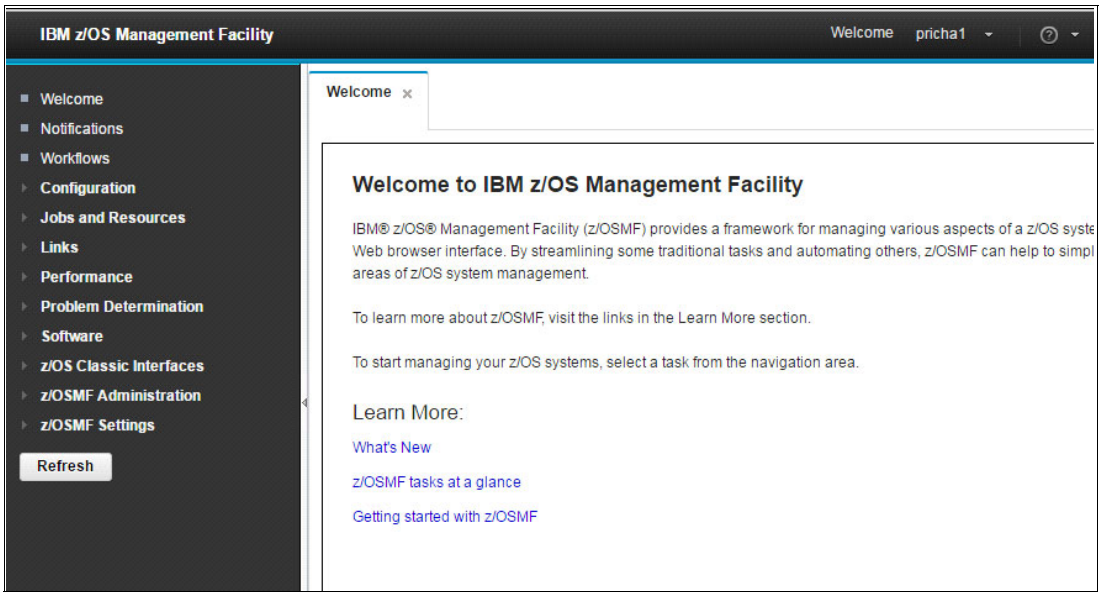

Figure 5-12 z/OSMF Welcome window after successful logon

5.4.3 Logging on to MVS Console

In this section, the first attempt uses RACF credentials only; the second uses the generated token code.

RACF credentials for MVS Console

If you attempt to logon to your z/OS console by using your RACF credentials, the attempt fails. You see the messages that are shown in Example 5-18.

Example 5-18 Log on to MVS Console failed messages

```
On your MVS console, in the logon status field:
IEE187I ENTER LOGON PARAMETERS -PASSWORD INCORRECT
LOGON DRICHAR  PASSWORD
GROUP          SECLABEL
```

```
On the console (SYSLOG,OPERLOG):
- 07.44.55 X9           IRR013I  VERIFICATION FAILED. INVALID PASSWORD
-  GIVEN.

In the AZF#IN00 server log:
ZFSTC:MFAA Version=2, MFAA Length=264, Application=A6PMCSX9, STC UserID=+CONSOL
ZFSTC:Password length=8, Passphrase length=0, New Password Length=0, New Passph
ZFSTC:authStateTransition mfaAuthTxn START (23AEE1E8)
ZFSTC:authStateTransition (23AEE1E8) TO txnResult=0x1 (pre-switch)
ZFSTC:AZF2202W In-band auth failed
ZFSTC:authStateTransition (23AEE1E8) TO txnResult=0x1 (post-switch)
ZFSTC:PC return code=0x8, reason code=0x0, abend code=0x0
```

Using generated token code for MVS Console

Instead of using the RACF password or passphrase to log on to the MVS Console, use the passcode you generated. The logon is successful and you see messages in the AZF#IN00 server log similar to the messages that are shown in Example 5-19.

Example 5-19 Successful logon to MVS Console by using a generated token code

```
AZFSTC:AZF2105I Authentication request (PC)
AZFSTC:MFAA Version=2, MFAA Length=264, Application=A6PMCSX9, STC UserID=+CONSOL
AZFSTC:Password length=8, Passphrase length=0, New Password Length=0, New Passph
AZFSTC:authStateTransition mfaAuthTxn START (23AEE1E8)
AZFSTC:authStateTransition (23AEE1E8) TO txnResult=0x1 (pre-switch)
AZFSTC:AZF2201I In-band auth success
AZFSTC:authStateTransition (23AEE1E8) TO txnResult=0x1 (post-switch)
AZFSTC:PC return code=0x0, reason code=0x0, abend code=0x0
```

5.5 Reregistering a user account

This section describes the steps that are used to clear the user's TOTP token and reregister the user.

After the user clears the Account from their iOS device, the user can reregister a new TOTP token with the server. At the end of the registration process, the user is ACTIVE, in `REGSTATE:PROVISIONED`, and can use their account.

Complete the following steps to reregister an account:

1. Deactivate the user for IBM TouchToken, as shown in Example 5-20.

   Example 5-20 Deactivating the user for IBM TouchToken

   ```
   ALU [Login ID] MFA(FACTOR(AZFTOTP1) NOACTIVE)
   ```

2. Delete the IBM TouchToken AZFTOTP1 tags that are associated with the account, as shown in Example 5-21.

   Example 5-21 Deleting the AZFTOTP1 tags associated with the user

   ```
   ALU [Login ID] MFA(FACTOR(AZFTOTP1) NOACTIVE NOTAGS)
   ```

3. Set the IBM TouchToken registration state for the user to OPEN (case-sensitive), as shown in Example 5-22.

Example 5-22 Setting registration state to OPEN

```
ALU [Login ID] MFA(FACTOR(AZFTOTP1) TAGS(REGSTATE:OPEN))
```

4. Instruct the user to start the IBM TouchToken for iOS application on the Apple device and delete the IBM TouchToken account. (Swipe left to delete the account).
5. Ensure that the user's Apple device has network connectivity to the IBM TouchToken registration server.
6. Instruct the user to open the URL of the IBM TouchToken registration server in Safari and log in with their z/OS user name and password. An IBM TouchToken account is created for the IBM TouchToken for iOS application.

Note: Consider advising the user to rename this account so that any system-specific information is removed.

7. Enter the command that is shown in Example 5-23 to display IBM MFA information for a user profile. After the IBM TouchToken account is set up on the Apple device, the REGSTATE changes to PROVISIONED.

Example 5-23 Displaying MFA status for user

```
LISTUSER [Login ID] MFA

MULTIFACTOR AUTHENTICATION INFORMATION:
---------------------------------------
FACTOR = AZFTOTP1
STATUS = INACTIVE
FACTOR TAGS =
REGSTATE:PROVISIONED
ALG:SHA384
KEYLABEL:AZF.AZFTOTP1.UserID.D095C000F3B861C7
WINDOW:1
NUMDIGITS:8
CVALUE:48682555
PERIOD:30
```

8. Instruct the user to start the IBM TouchToken for iOS application on the Apple device.
9. Instruct the user to tap the new IBM TouchToken account.

   When prompted, the user must supply their Apple TouchID fingerprint. If successful, the IBM TouchToken OTP token code is displayed.

   The user must now use this OTP token code to log on to the z/OS system.

5.5.1 Tag considerations

There are considerations to be aware of when tag values are changed.

As shown in Example 5-24 on page 96, the command alters the value of the WINDOW tag. The command is run successfully.

Example 5-24 Changing the WINDOW tag

```
ALU LogonID MFA(FACTOR(AZFTOTP1) TAGS(WINDOW:2))
```

As shown in Example 5-25, the command alters the value of the PERIOD tag.

Example 5-25 Changing the PERIOD tag

```
ALU LogonID MFA(FACTOR(AZFTOTP1) TAGS(PERIOD:30))
```

The command fails because ICH21051I IBM MFA detected an error in the value of tag PERIOD with the following message:

```
AZF1110I PERIOD must be 15, 30, or 60 (or REGSTATE is not OPEN)
```

List the user's MFA information by using the command that is shown in Example 5-26. The response also is shown.

Example 5-26 LIsting MFA information for user

```
LISTUSER LogonID MFA

MULTIFACTOR AUTHENTICATION INFORMATION:
---------------------------------------
 PASSWORD FALLBACK IS ALLOWED
 FACTOR = AZFTOTP1
   STATUS = ACTIVE
   FACTOR TAGS =
     REGSTATE:PROVISIONED
     KEYLABEL:AZF.DRICHAR.D16117OFC18FC758
     ALG:SHA256
     CVALUE:24581523
     NUMDIGITS:8
     PERIOD:60
     WINDOW:2
```

Although the PERIOD TAG cannot be changed, the WINDOW TAG can be changed.

The user must be in REGSTATE:OPEN for the PERIOD (and certain other tags) to be modified by the administrator. After the user is in REGSTATE:PROVISIONED (a state set by the TOTP registration server), *any* modification of the PERIOD setting causes the TOTP token to become unusable by the user because the server and client produces different token codes.

You must alter the user to NOTAGS. The ALU commands (see Example 5-27) are then successful.

Example 5-27 Modifying TAGs

```
ALU LogonID MFA(FACTOR(AZFTOTP1) NOTAGS PWFALLBACK
...
ALU DRICHAR MFA(FACTOR(AZFTOTP1) TAGS(WINDOW:2))
...
ALU DRICHAR MFA(FACTOR(AZFTOTP1) TAGS(PERIOD:30))
...
ALU DRICHAR MFA(FACTOR(AZFTOTP1) ACTIVE PWFALLBACK
```

Note: If you deleted all tags, you must delete the account on the iOS device and reregister the account.

In another instance of the PERIOD tag (as shown in Example 5-28), MFA information in which the PERIOD and WINDOW tags the system default values is used.

Example 5-28 Default tags listing

```
MULTIFACTOR AUTHENTICATION INFORMATION:
---------------------------------------
  PASSWORD FALLBACK IS NOT ALLOWED
  FACTOR = AZFTOTP1
    STATUS = ACTIVE
    FACTOR TAGS =
      REGSTATE:PROVISIONED
      KEYLABEL:AZF.DRICHAR.D167A1CC4BB70654
      ALG:SHA256
      CVALUE:24581666
      NUMDIGITS:8
      PERIOD:60
      WINDOW:2
```

If the **ALU** command that is shown in Example 5-29 is entered to delete the PERIOD tag, the command runs successfully. The MFA segment also is shown and no longer includes the PERIOD tag.

Example 5-29 Removing the PERIOD tag

```
ALU LogonID MFA(FACTOR(AZFTOTP1) DELTAGS(PERIOD))

MULTIFACTOR AUTHENTICATION INFORMATION:
 ---------------------------------------
   PASSWORD FALLBACK IS NOT ALLOWED
   FACTOR = AZFTOTP1
     STATUS = ACTIVE
     FACTOR TAGS =
       REGSTATE:PROVISIONED
       KEYLABEL:AZF.DRICHAR.D167A1CC4BB70654
       ALG:SHA256
       CVALUE:24581667
       NUMDIGITS:8
       WINDOW:2
```

Deleting the PERIOD tag means that the user can no longer log on with any generated token. If the user attempts to log on, the messages from the SYSLOG and AZF server that are shown in Example 5-30 are issued.

Example 5-30 SYSLOG and AZF messages

```
SYSLOG Messages:

ICH408I USER(DRICHAR ) GROUP(OMVSGRP ) NAME(DOMINIQUE RICHARD   ) 313
  LOGON/JOB INITIATION - MULTIFACTOR AUTHENTICATION UNAVAILABLE

In the AZF server:

AZFSTC:AZF2105I Authentication request (PC)
AZFSTC:MFAA Version=2, MFAA Length=264, Application=IZUDFLT , STC UserID=IZUSVR
AZFSTC:Password length=8, Passphrase length=0, New Password Length=0, New Passph
AZFSTC:authStateTransition mfaAuthTxn START (23AEE080)
AZFTOTP:AZF4110E AZFTOTP User object validation failed (DRICHAR, 67616)
AZFSTC:authStateTransition (23AEE080) TO txnResult=0x1 (pre-switch)
AZFSTC:AZF2202W In-band auth failed
AZFSTC:authStateTransition (23AEE080) TO txnResult=0x1 (post-switch)
AZFSTC:PC return code=0x8, reason code=0x1, abend code=0x0
```

The reason for these messages is that after the user is in REGSTATE:PROVISIONED (a state that is set by the TOTP registration server), any modification of the PERIOD setting causes the TOTP token to become unusable by the user because the server and client produce different token codes.

5.5.2 AZFPTKT1 factor

If a strong factor is used, such as AZFTOTP1 or AZFSIDP1, you might also specify AZFPTKT1 for the user. However, you cannot specify AZFPTKT1 alone as a single factor because RACF successfully evaluates PassTickets without specifying MFA data for a user.

PassTickets cannot be used if you enabled TOTP factor, unless you also enable the PassTicket MFA factor, as shown in Example 5-31.

Example 5-31 Enabling the PassTicket factor for a user

```
ALU PRICHA1 MFA(FACTOR(AZFPTKT1)ACTIVE TAGS(MFAFIRST:N))
```

Therefore, even if AZFPTKT1 is not enabled or inactive, PassTickets do not work for users with a TOTP factor, as shown in the following example:

- AZFTOTP1 factor with AZFPTKT1 factor => PassTicket ok
- AZFTOTP1 factor with no AZFPTKT1 factor (or inactive) => KO

If you remove the TOTP factor (inactive) and keep the AZFPTKT1 factor, you also can log on by using the PassTicket. However, but you receive the messages that are shown in Example 5-32.

Example 5-32 Logon messages

```
Messages on the TSO screen

ICH70008I IBM MFA Message:
no active factor
ICH70001I MFAUSER LAST ACCESS AT 13:55:21 ON THURSDAY, SEPTEMBER 22, 2016
IKJ56455I MFAUSER LOGON IN PROGRESS AT 13:56:30 ON SEPTEMBER 22, 2016

Messages in the SYSLOG:
AZF2211E Auth preparation failed, cannot evaluate
$HASP100 MFAUSER ON TSOINRDR
$HASP373 MFAUSER STARTED
```

6

MFA application selection and PassTicket support

IBM Multi-Factor Authentication (MFA) can be bypassed for specific applications and specific application users, which allows IBM MFA access for specific applications and specific application users only. It also enables setting a default IBM MFA bypass profile for applications that are not otherwise allowed or bypassed.

For each application that users can access with a PassTicket, at least one profile in the PTKTDATA class must be created. The profile associates a secret, secured sign-on application key with a particular application on a particular system.

This chapter includes the following topics:

6.1 Selective MFA bypass processing for applications

The MFA application selection enhancements are in the following areas:

- Selective MFA bypass by using APPLID profiles
- Selective MFA bypass by using User ID profiles

If a user was selected to use MFA credentials to logon before IBM MFA for z/OS V1.1, this type of enforcement can cause issues for applications that include authentication properties that do not align with MFA's expectations, which prevent MFA from working effectively.

The following application properties can cause issues for MFA:

- No phrase support

  Some of the MFA authenticators can be longer than eight characters, which are not accommodated by the application.

- No password change field

  MFA can use the password change field to change an RSA SecurID PIN during logon processing.

- Use of PassTicket authenticators

  This option not supported by MFA.

- Replay of passwords

  Some MFA credentials are different at every logon and cannot be displayed.

In addition to application properties, the local security policy might stipulate conditions whereby MFA is enforced or not required. The following circumstances can influence the decisions:

- The role of the application and its sensitivity.
- Development or test applications of low importance with non-sensitive data.
- User training applications with set User IDs and passwords.
- Emergency systems, such as applications that assist in a recovery in a disaster recovery situation or severely affected system.
- Test applications that are used to test specific system software changes after maintenance was applied.

> **Note:** If you bypass IBM MFA, the application users must use their RACF password to log on.

6.1.1 Considerations for bypassing MFA

The following high-level scenarios are examples of bypassing IBM MFA for specific applications or allowing IBM MFA access for specific applications:

- The application provides the RACF application name and you know the APPLNAME value.

  In this case, IBM MFA generates a profile of the name `MFABYPASS.APPL.applname` and tests the user's access against this profile in the MFADEF class. If the returned access is NONE, IBM MFA authenticates the credentials as IBM MFA credentials. If the returned access is READ or higher, IBM MFA authenticates the credentials as valid RACF credentials (password or passphrase, as appropriate). No further profile checks are made.

- The application does not provide the RACF application name, but the authentication is performed by an address space, such as STC, that is running with a defined user ID.

  In this case, IBM MFA generates a profile of the name `MFABYPASS.USERID.STCUSERID` and tests the user's access against this profile in the MFADEF class. If the returned access is NONE, IBM MFA authenticates the credentials as IBM MFA credentials. If the returned access is READ or better, IBM MFA authenticates the credentials as valid RACF credentials (password or passphrase, as appropriate). No further profile checks are made.

- The application does not provide the RACF application name and the authentication is performed by an address space, such as STC, that is not running with a defined user ID or it is occurring during address space creation.

  In this case, IBM MFA generates a profile of the name `MFABYPASS.DEFAULT` and tests the user's access against this profile in the MFADEF class. If the returned access is NONE, IBM MFA authenticates the credentials as IBM MFA credentials. If the returned access is READ or better, IBM MFA authenticates the credentials as valid RACF credentials (password or passphrase, as appropriate). No further profile checks are made.

Note: It is recommended that you define profiles `MFABYPASS.APPL.*` and `MFABYPASS.USERID.*` with an access level of UACC(NONE) and no access list to ensure that no unintended bypasses of IBM MFA occur because the profiles are in the MFADEF class.

6.2 Bypassing IBM MFA for applications by application name

You can bypass IBM MFA for specific applications and specific application users. After you bypass IBM MFA, the application users must use their RACF credentials to log on.

You can determine the relevant information that is issued by any particular authentication request that is made by an IBM MFA user in the MFA server log, provided trace level 2 is activated.

The IBM MFA started task's SYSPRINT contains lines that show the application name and User ID values that are supplied explicitly and implicitly. This information can be used to guide you in defining profiles.

The output from a level 2 trace is shown in Example 6-1 on page 104. From the sample messages, we can determine that the TSO application name was `A6PTXO` (no STC User ID), the z/OSMF application name was `IZUDFLT` (STC User ID IZUSVR), and MVS console application name was `A6PMCSX9` (because we use the SMCS console) and STC User ID is `+CONSOLE`.

Example 6-1 Sample trace messages output

```
AZFSTC:AZF2105I Authentication request (PC)
AZFSTC:MFAA Version=2, MFAA Length=264, Application=A6PTX0  , STC UserID=

AZFSTC:AZF2105I Authentication request (PC)
AZFSTC:MFAA Version=2, MFAA Length=264, Application=IZUDFLT , STC UserID=IZUSVR

AZFSTC:AZF2105I Authentication request (PC)
AZFSTC:MFAA Version=2, MFAA Length=264, Application=A6PMCSX9, STC UserID=+CONSOLE
```

6.3 Bypassing IBM MFA for applications by User ID

In this scenario, a User ID (MFAUSE1) with an MFA TOTP factor is registered as an account in their iOS device (see the RACF and MFA that is shown in Example 6-2).

Example 6-2 Listing MFAUSER RACF and MFA information

```
LU MFAUSE1 MFA
USER=MFAUSE1 NAME=JOHN OWNER=MFAUSER CREATED=00.031
 DEFAULT-GROUP=SYS1     PASSDATE=16.281 PASS-INTERVAL= 60 PHRASEDATE=13.330
 PASSWORD ENVELOPED=YES
 PHRASE ENVELOPED=YES
 ATTRIBUTES=SPECIAL
 ATTRIBUTES=PASSPHRASE
 REVOKE DATE=NONE   RESUME DATE=NONE
...
...
 LAST-ACCESS=16.294/08:04:00 MULTIFACTOR AUTHENTICATION INFORMATION:
---------------------------------------
  PASSWORD FALLBACK IS NOT ALLOWED
  FACTOR = AZFTOTP1
    STATUS = ACTIVE
FACTOR TAGS =
      REGSTATE:PROVISIONED
      KEYLABEL:AZF.DRICHAR.D171603F29653359
      ALG:SHA256
      CVALUE:24611964
      NUMDIGITS:8
      PERIOD:60
      WINDOW:10
***
```

The RACF definition that is shown in Example 6-3 bypasses IBM MFA for the A6PTX0 application (TSO in our scenario) for all users who are granted at least READ access to a profile in the MFADEF class for the application.

Example 6-3 RACF definition to bypass MFA for the A6PTX0 TSO application

```
RDEFINE MFADEF MFABYPASS.APPL.A6PTX0 UACC(READ)
```

The commands that are shown in Example 6-4 bypass the A6PTX0 application for user MFAUSE1 only.

Example 6-4 Bypassing for specific user

```
RDEFINE MFADEF MFABYPASS.APPL.A6PTX0 UACC(NONE)
PERMIT MFABYPASS.APPL.A6PTX0 CLASS(MFADEF) ID(MFAUSE1) ACCESS(READ)
```

The commands that are shown in Example 6-5 bypass IBM MFA for all applications, except the A6PTX0 application that is identified with a profile in the MFADEF class with access NONE.

Example 6-5 Bypassing MFA for all applications except A6PTX0

```
RDEFINE MFADEF MFABYPASS.APPL.* UACC(READ)
RDEFINE MFADEF MFABYPASS.APPL.A6PTX0 UACC(NONE)
```

Suppose the User ID MFAUSE1 can log on to the z/OS applications only by providing the generated TOTP passcode. If this user enters only their RACF credentials, the logon fails because the user is a defined MFA user for TOTP.

To allow User ID MFAUSE1 to log on to the A6PTX0 (TSO) application, MFA must be bypassed. MFAUSE1 can then use RACF credentials instead of the TOTP passcode. The commands that are used to achieve this bypass are shown in Example 6-6. The READ access bypasses the MFA authentication for the application.

Example 6-6 Allowing MFAUSE1 to bypass MFA for application A6PTX0

```
RDEFINE MFADEF MFABYPASS.APPL.A6PTX0 UACC(NONE)
PERMIT MFABYPASS.APPL.A6PTX0  CLASS(MFADEF) ID(MFAUSE1) ACCESS(READ)
SETROPTS REFRESH RACLIST(MFADEF)
```

If you list the profile (see Example 6-7) you see that the MFAUSE1 includes READ access to the bypass profile.

Example 6-7 Bypass profile listing

```
RLIST MFADEF MFABYPASS.APPL.A6PTX0 ALL
CLASS      NAME
-----      ----
MFADEF     MFABYPASS.APPL.A6PTX0

LEVEL  OWNER      UNIVERSAL ACCESS  YOUR ACCESS  WARNING
-----  --------   ----------------  -----------  -------
00    MFAUSER         NONE               READ    NO


 CREATION DATE  LAST REFERENCE DATE  LAST CHANGE DATE
  (DAY) (YEAR)       (DAY) (YEAR)      (DAY) (YEAR)
 -------------  -------------------  ----------------
   264    16          264    16         264    16
ALTER COUNT   CONTROL COUNT   UPDATE COUNT   READ COUNT
 -----------   -------------   ------------   ----------
   000000         000000         000000         000000

 USER      ACCESS   ACCESS COUNT
 ----      ------   ------ -----
```

```
MFAUSER   READ        000000
 MFAUSE1 READ        000000

    ID     ACCESS  ACCESS COUNT  CLASS                    ENTITY  NAME
 -------- -------  ------------ -------- ------------------------------
 NO ENTRIES IN CONDITIONAL ACCESS LIST
 ***
```

The user MFAUSE1 now has READ access to profile `MFABYPASS.APPL.A6PTX0` in class MFADEF and can log on to the A6PTX0 application by using their RACF credentials. The bypass is shown in the AZF#INF00 server log (see Example 6-8).

Example 6-8 Confirmation of bypass in AZF#IN00 log

```
AZFSTC:AZF2105I Authentication request (PC)
AZFSTC:MFAA Version=2, MFAA Length=264, Application=A6PTX0  , STC UserID=
AZFSTC:Password length=8, Passphrase length=0, New Password Length=0, New Passph
AZFSTC:MFA bypassed because of access 4 to profile MFABYPASS.APPL.A6PTX0 in class
MFADEF
AZFSTC:PC return code=0x8, reason code=0x5, abend code=0x0
```

However, if the MFAUSE1 user attempts to log on to any other application by using their RACF password, the attempt fails because no other bypass profile was created for the other applications that allows the MFAUSE User ID READ access. The z/OSMF bypass profile is shown in Example 6-9.

Example 6-9 z/OSMF bypass profile listing

```
RLIST MFADEF MFABYPASS.APPL.IZUDFLT ALL
CLASS      NAME
-----      ----
MFADEF     MFABYPASS.APPL.IZUDFLT

LEVEL  OWNER      UNIVERSAL ACCESS  YOUR ACCESS  WARNING
-----  --------   ----------------  -----------  -------
 00    PRICHAR         NONE               READ    NO

INSTALLATION DATA
-----------------
NONE
...
CREATION DATE  LAST REFERENCE DATE  LAST CHANGE DATE
  (DAY) (YEAR)       (DAY) (YEAR)      (DAY) (YEAR)
 -------------  ------------------  ----------------
   264    16          264    16         264    16

 ALTER COUNT   CONTROL COUNT   UPDATE COUNT   READ COUNT
 -----------   -------------   ------------   ----------
   000000         000000         000000         000000

 USER      ACCESS   ACCESS COUNT
 ----      ------   ------ -----
 PRICHAR   READ        000000

    ID     ACCESS  ACCESS COUNT  CLASS                   ENTITY  NAME
 -------- -------  ------------ -------- -----------------------------
 NO ENTRIES IN CONDITIONAL ACCESS LIST
 ***
```

The MFAUSE1 user is not defined in the access list of this profile and the profile includes a UACC(NONE). Therefore, if the MFAUSE1 user attempts to log on to z/OSMF (application IZUDFLT) and provide an RACF password, the attempts fails. Messages in the SYSLOG and AZF#IN00 indicate the failure, as shown in Example 6-10.

Example 6-10 Messages that indicate failure for MFAUSE1 to log on to z/OSMF

```
SYSLOG messages:
ICH408I USER(MFAUSE1 ) GROUP(SYS1    ) NAME(JOHN            ) 174
  LOGON/JOB INITIATION - MULTIFACTOR AUTHENTICATION FAILURE
IRR013I  VERIFICATION FAILED. INVALID PASSWORD GIVEN.

AZF#IN00 server messages:
AZFSTC:AZF2105I Authentication request (PC)
AZFSTC:MFAA Version=2, MFAA Length=264, Application=IZUDFLT , STC UserID=IZUSVR
AZFSTC:Password length=8, Passphrase length=0, New Password Length=0, New Passph
AZFSTC:authStateTransition weakAuthTxn START (23B02228)
AZFPTKT:AZF7012I: Applying user-specific eval policy (mfaRequired=0)
AZFSTC:authStateTransition mfaAuthTxn START (23AEE170)
AZFSTC:authStateTransition (23AEE170) TO txnResult=0x1 (pre-switch)
AZFSTC:AZF2202W In-band auth failed
AZFSTC:authStateTransition (23AEE170) TO txnResult=0x1 (post-switch)
AZFSTC:PC return code=0x8, reason code=0x0, abend code=0x0
```

6.4 PassTicket support

Support for PassTickets was introduced in IBM MFA for z/OS V1.1. Circumstances can arise in which you choose to bypass the PassTicket support. In this section, we describe how to bypass this support.

The RACF PassTicket is a one-time only password that is generated by a requesting product or function. It is an alternative to the RACF password and password phrase that removes the need to send RACF passwords and password phrases across the network in clear text.

You can configure IBM MFA to allow the use of a PassTicket only after a successful IBM MFA logon, or to use only the PassTicket and not an IBM MFA logon.

Complete the following steps:

1. Use the **RDEFINE** (**RDEF) command** to define an MFADEF class profile that is named FACTOR.AZFPTKT1 and the **RLIST** command to verify it, as shown in Example 6-11.

   Example 6-11 Defining and verifying MFADEF class profile for PassTicket

   ```
   RDEF MFADEF FACTOR.AZFPTKT1

   RLIST MFADEF FACTOR.AZFPTKT1 MFA
   ```

2. Define and verify a FACILITY class profile that is named `IRR.RFACTOR.MFADEF.AZFPTKT1`, as shown in Example 6-12.

Example 6-12 Defining and verifying a FACILITY class profile

```
RDEF FACILITY IRR.RFACTOR.MFADEF.AZFPTKT1

RLIST FACILITY IRR.RFACTOR.MFADEF.AZFPTKT1
```

3. Authorize the administrators that run the panels to the `IRR.RFACTOR.MFADEF.AZFPTKT1` profile. Choose the access that is listed in Table 6-1.

Table 6-1 PassTicket levels of access

Permission	Access
READ	Can view configuration options, but cannot update, create, or delete PassTicket parameters.
UPDATE	Can view and update configuration options, but cannot create or delete PassTicket parameters.
CONTROL	Can view configuration options, but cannot update, create, or delete PassTicket parameters.
ALTER	Can view configuration update, create, or delete PassTicket configuration options.

In our controlled environment, we chose ALTER access and defined and activated the access by using the commands that are shown in Example 6-13.

Example 6-13 Defining and activating ALTER authority to PassTicket to Facility class

```
PERMIT IRR.RFACTOR.MFADEF.AZFPTKT1 CLASS(FACILITY) ID(SYS1) +
     ACCESS(ALTER)
SETROPTS RACLIST(FACILITY) REFRESH
```

4. Run the `AZFEXEC REXX` procedure. The ISPF MFA Plug-in Administration information that is shown in Example 6-14 appears. Select the **PT** option.

Example 6-14 MFA Plug-in Administration

```
IBM Multi-Factor Authentication for z/OS
                              Plug-in Administration
Command ===> PT

                           Configure Plug-ins

                          PT  AZFPTKT1 Plug-in
                          S   AZFSIDP1 Plug-in
                          T   AZFTOTP1 Plug-in
```

The Multi-Factor Authentication AZFPTKT1 Plug-in Attributes are displayed (see Example 6-15 on page 109). By using the panel, the plug-in's factor-wide attributes in the associated MFADEF RACF Class profile can be created, edited, viewed, and deleted.

To enter the attributes, the following levels of authority to the `IRR.RFACTOR.MFADEF.AZFPTKT1` profile in the RACF FACILITY Class are needed:

- Create Attributes: ALTER
- Modify Attributes: UPDATE

– View Attributes: READ
– Delete Attributes: ALTER

Example 6-15 AZFPTKT1 Plug-in Attributes panel

```
IBM Multi-Factor Authentication for z/OS
                          AZFPTKT1 Plug-in Attributes
Command ===>

User Defaults

  Require MFA Logon prior to PassTicket Evaluation  Y

  PassTicket Evaluation Window . . . . . . . . . .  30

Plug-In

  Initial Trace Level . . . . 3
```

In our controlled environment we chose the following values:

– Require MFA Logon prior to PassTicket Evaluation

  Enter `Y` to require successful MFA authentication before PassTickets can be evaluated.

– PassTicket Evaluation Window

  Length of time (in seconds) that PassTickets can be used to authenticate after a successful MFA authentication. Valid values are 30 - 86400.

– Trace Level

  The value that is used for tracing events within the AZFPTKT1 registration server. Values can be specified 0 - 3, where the higher number increases the level of verbosity.

5. Activate users for PassTicket by using the **ALU** command, as shown in Example 6-16.

Example 6-16 Activating the user for PassTicket

```
ALU LOGIN ID MFA(FACTOR(AZFPTKT1)
ACTIVE TAGS(WINDOW:numseconds MFAFIRST:Y|N))
```

As shown in Example 6-16, the following parameters are used:

– Login ID is the z/OS user name.
– ACTIVE activates the AZFPTKT1 authenticator for the user ID.
– WINDOW sets the evaluation window as a number of seconds.
– MFAFIRST specifies whether to require a successful IBM MFA logon before the PassTicket being evaluated. The possible values are Y and N, and uppercase is required.

If you set MFAFIRST or WINDOW for a user, the default setting is overridden.

To return a user to the default settings, use the command that is shown in Example 6-17.

Example 6-17 Returning a user to default settings

```
ALU LOGIN ID MFA(FACTOR(AZFPTKT1) DELTAGS(MFAFIRST WINDOW))
```

A sample job to set up PassTicket support and display the MFA information is shown in Example 6-18.

Example 6-18 Sample job for PassTicket support

```
//ALTUSER  EXEC PGM=IKJEFT01,DYNAMNBR=30
//SYSPROC  DD DISP=SHR,DSN=SYS1.SBPXEXEC
//SYSLBC  DD DSN=SYS1.BRODCAST,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD  *
 ALU MFAUSER MFA(FACTOR(AZFPTKT1) +
 ACTIVE TAGS(MFAFIRST:N))
/*
... Subsequent command and display:
LISTUSER MFAUSER MFA
...
MULTIFACTOR AUTHENTICATION INFORMATION:
---------------------------------------
  PASSWORD FALLBACK IS ALLOWED
  FACTOR = AZFTOTP1
    STATUS = ACTIVE
    FACTOR TAGS =
      REGSTATE:PROVISIONED
      KEYLABEL:AZF.MFAUSER.D15FD378FC229063
      ALG:SHA256
      CVALUE:24592795
      NUMDIGITS:8
      PERIOD:60
      WINDOW:10
  FACTOR = AZFPTKT1
    STATUS = ACTIVE
    FACTOR TAGS =
      MFAFIRST:N
***
```

6. Check that the User ID of the MFA server AZF#IN00 (AZFSTC) includes READ access to the `PTKTDATA IRRPTAUTH.RACF_APPLICATION_NAME.USERID` profile. The MFAUSER should log on by using PassTickets. For each application that users can access by using the PassTicket, you must create at least one profile in the PTKTDATA class. The profile associates a secret secured sign-on application key with a particular application on a particular system.

   We defined the TSO and z/OSMF applications in our controlled environment, as shown in Example 6-19.

   Example 6-19 Commands to set up the PassTicket applications

```
SETROPTS CLASSACT(PTKTDATA)
SETROPTS RACLIST(PTKTDATA)
RDEFINE PTKTDATA IRRPTAUTH.A6PTX0.* UACC(NONE)
RDEFINE PTKTDATA IRRPTAUTH.IZUDFLT.* UACC(NONE)
PE IRRPTAUTH.A6PTX0.* CLASS(PTKTDATA) ID(AZFSTC)  ACC(READ)
PE IRRPTAUTH.A6PTX0.PRICHA1 CLASS(PTKTDATA) ID(AZFSTC) ACC(UPDATE)
PE IRRPTAUTH.IZUDFLT.PRICHA1 CLASS(PTKTDATA) ID(AZFSTC) ACC(UPDATE)
PE IRRPTAUTH.IZUDFLT.* CLASS(PTKTDATA) ID(AZFSTC)  ACC(READ)
SETROPTS RACLIST(PTKTDATA) REFRESH
```

A PassTicket can now be defined for user MFAUSER to log on to applications TSO (A6PTX0) and z/OSMF (IZUDFLT).

When user MFAUSER logs on to z/OSMF with the new PassTicket, the messages (as shown in Example 6-20) are in the server log that shows that the logon was granted as a result of the PassTicket evaluation.

Example 6-20 Verification of logon by using a PassTicket

```
AZFSTC:AZF2105I Authentication request (PC)
AZFSTC:MFAA Version=2, MFAA Length=264, Application=IZUDFLT , STC UserID=IZUSVR
AZFSTC:Password length=8, Passphrase length=0, New Password Length=0, New Passph
AZFSTC:authStateTransition weakAuthTxn START (23B02228)
AZFPTKT:AZF7011I: Txn is a candidate for PassTicket eval
AZFPTKT:AZF7012I: Applying user-specific eval policy (mfaRequired=0)
AZFPTKT:AZF7005I: Result of PassTicket eval: safrc:0, racfrc: 0, racfrsn: 0
AZFSTC:PC return code=0x0, reason code=0x1, abend code=0x0
```

7

Operational information and FAQs

In this chapter, we describe operational information that is related to the associated IBM Multi-Factor Authentication (MFA) servers that are all started tasks on z/OS.

This chapter also includes information that is based on frequently asked questions (FAQs) to help give you more information about MFA.

This chapter includes the following topics:

7.1 Operation commands

In this section, we describe the operator commands that are used to manage the MFA servers.

7.1.1 Starting and stopping the MFA servers

The IBM TouchToken registration server started task supports the start operation at run time.

To start the IBM TouchToken registration server started task, enter the following operator command:

```
S AZF#IN01
```

To stop the IBM TouchToken registration server started task, enter the following operator command:

```
P AZF#IN01
```

To start the MFA started task, enter the following operator command:

```
S AZF#IN00
```

To stop the MFA started task, enter the following operator command:

```
P AZF#IN00
```

Note: Ensure that your automation software includes all the activities to support your MFA environment.

7.1.2 Modifying MFA component trace levels

The IBM MFA started task supports modifying trace levels on a per-component basis at run time. The available trace levels are listed Table 7-1.

Table 7-1 MFA trace level information

Trace level	Description
0	Only standard and unconditional messages.
1	All output from level 0 plus major items of interest.
2	All output form level 1 plus lesser items of interest.
3	All trace information (Verbose).

To change trace levels on a per-component basis, issue a Modify (F) z/OS system command. The command format is shown in Example 7-1.

Example 7-1 Modify command format

```
F <STC Job Name>,<Component> SET TRACE LEVEL <Trace Level>
```

The <STC Job Name> represents the started task. The component can be one of the following three literal values:

- AZFSIDP1 represents the AZFSIDP1 authenticator that supports RSA SecurID.
- AZFTOTP1 represents the AZFTOTP1 authenticator that supports IBM TouchToken.
- AZFPTKT1 represents the AZFPTKT1 authenticator that supports PassTickets.

The modify command can be issued against any of the MFA-related started tasks, as shown in Example 7-2.

Example 7-2 Sample modify command

```
F AZF#INOO,AZFTOTP1 SET TRACE LEVEL 1
```

7.1.3 Determining relevant authentication information

You can determine the relevant information that is issued by an authentication request. To determine the relevant information that is issued by any specific authentication request that is made by an IBM MFA user, issue the modify command (shown in Example 7-3) and then attempt or retry an authentication.

Example 7-3 Modify command to set trace level 2

```
F <MFA_STC_Job_Name>,STC SET TRACE LEVEL 2
```

Search the logs for entries that begin with MFAA, similar to the message that is shown in Example 7-4.

Example 7-4 MFAA-related message

```
MFAA Version=2, MFAA Length=264, Application=TESTAPP, STC UserID=TSTUSR
```

The IBM MFA started task's SYSPRINT contains lines that show the application name and User ID values that are supplied explicitly and implicitly by the issuer of the RACROUTE REQUEST=VERIFY call, which you can use to guide you in defining profiles (see Example 7-5).

Example 7-5 MFA messages that are related to the failure

```
AZFSTC:PC return code=0x8, reason code=0x5, abend code=0x0
AZFSTC:AZF2105I Authentication request (PC)
AZFSTC:MFAA Version=2, MFAA Length=264, Application=A6PTX0  , STC UserID=
AZFSTC:Password length=8, Passphrase length=0, New Password Length=0, New Passph
AZFSTC:authStateTransition mfaAuthTxn START (23AEE1E8)
AZFSTC:authStateTransition (23AEE1E8) TO txnResult=0x1 (pre-switch)
AZFSTC:AZF2202W In-band auth failed
AZFSTC:authStateTransition (23AEE1E8) TO txnResult=0x1 (post-switch)
AZFSTC:PC return code=0x8, reason code=0x0, abend code=0x0
```

7.2 FAQs

This section includes the following FAQs and answers that cover practical situations that are related to aspects MFA usage:

- Question: *Is it possible to register a single user on the same server with two or more devices (assuming that I have an iPhone and an iPad with my own fingerprints registered, for instance), and I forgot my iPhone at home?*

  Answer: At the time of this writing, this type of registration cannot be done. You cannot register a single user on the same server with two or more devices.

- Question: *What if my iOS device is not connected to the same network as my z/OS system or the application to which I want to connect?*

  Answer: You must be within the same network if you want to register an account (z/OS User ID) on your iOS device. The registration process starts by starting the registration server that runs on your z/OS system on port 6789 (AZF#IN01).

  However, after your account is registered, you no longer need any network connectivity to your z/OS host or application to generate the TOTP passcode. The process of generating the token is stand-alone and runs solely on your iOS device by using the secret, which is stored on the device.

- Question: *What happens if I delete (swipe left to delete the account) a user account on my iOS device? Is it de-registered in the MFA registration server? Can I reregister it?*

  Answer: You are still registered in the MFA registration server. If you display the RACF MFA information, you see a display that is similar to the display that is shown in Example 7-6. You are still showing as ACTIVE.

Example 7-6 MFA segment information display

```
MULTIFACTOR AUTHENTICATION INFORMATION:
---------------------------------------
  PASSWORD FALLBACK IS NOT ALLOWED
  FACTOR = AZFTOTP1
    STATUS = ACTIVE
FACTOR TAGS =
      REGSTATE:PROVISIONED
      KEYLABEL:AZF.DRICHAR.D171603F29653359
      ALG:SHA256
      CVALUE:24621660
      NUMDIGITS:8
      PERIOD:60
      WINDOW:10
```

  However, if you attempt to reregister the User ID on your iOS device (as it is no longer in the list of your users), you receive an error message on the iOS device and in the MFA server log, as shown in Example 7-7.

Example 7-7 Reregistration error messages

```
Message on iOS device
An error has occurred
AZF5171E Authentication failed.

Messages in MFA server log
AZFSTC:AZF2105I Authentication request (PC)
AZFSTC:MFAA Version=2, MFAA Length=264, Application=AZFTOTPW, STC UserID=AZFSTC
```

```
AZFSTC:Password length=8, Passphrase length=0, New Password Length=0, New
Passph
AZFSTC:authStateTransition mfaAuthTxn START (23AEE168)
AZFTOTP:AZF4110E AZFTOTP User object validation failed (MFAUSER, 67612)
AZFSTC:authStateTransition (23AEE168) TO txnResult=0x1 (pre-switch)
AZFSTC:AZF2202W In-band auth failed
AZFSTC:authStateTransition (23AEE168) TO txnResult=0x1 (post-switch)
AZFSTC:PC return code=0x8, reason code=0x1, abend code=0x0
```

Explanation: AZFTOTP1 must be marked INACTIVE for registration. The registration server runs a **RACROUTE REQUEST=VERIFY** command to check the user's password (as entered in the app). If AZFTOTP1 is ACTIVE at the time that password check occurs, it fails because AZF catches the RACROUTE and attempts to check the user's AZFTOTP token (which they do not have and did not enter).

Solution: Reregister by following the steps that are shown in Example 7-8 to provide a sample solution. Check your local procedures.

Example 7-8 Reregistration steps

```
Deactivate MFA for the user's Logonid by issuing the command:
ALU LogonID MFA(FACTOR(AZFTOTP1) NOACTIVE
Remove the user's tags and specify no password fallback by issuing the command:
ALU LogonID MFA(FACTOR(AZFTOTP1) NOTAGS NOPWFALLBACK
Change the user's REGSTATE to OPEN to allow registration
ALU LogonID MFA(FACTOR(AZFTOTP1) TAGS(REGSTATE:OPEN))
Delete the account in the iOS device (swipe left)
Register the account by using the iOS device
```

- Question: *Can I change the TOTP token name?*

  Answer: Configuration data for IBM TouchToken is stored in the RACF database. The IBM TouchToken configuration data includes settings that are related to the AZFTOTP1 authentication load module and the IBM TouchToken registration server. If you change the PKCS#11 Token Name in a running AZFTOTP1 environment, PROVISIONED user accounts fail and you must reregister the users.

- Question: *What happens if I unregister a user account on my MFA registration server? What should I do with my registered account on my iOS device?*

  Answer: Because you no longer have a TOTP1 factor on the MFA segment for the user, you cannot use TOTP passcodes to log on to your z/OS. You must use your regular RACF/SAF credentials; otherwise, any passcode you enter is treated as a RACF password and evaluated as such. You can delete (swipe left, and press **delete** to delete) the account entry on your iOS device.

- Question: *What happens if I try to register a user account on my iOS device and the user's password that I enter in the iOS register account expired in RACF?*

  Answer: You receive an error message (see Example 7-9) on the iOS device and you must first log on to z/OS and change your password. As of this writing, no function is available to recognize an expired password and prompt you to change it during the process to register the account on the device.

Example 7-9 Log on error message on iOS device

```
An error has occurred
AZF5171E Authentication failed.
```

- Question: *Why would I need to reregister a user account?*

  Answer: You often do not need to reregister a user for IBM TouchToken unless there is a problem with the iOS device or the security of the shared secret was endangered.

- Question: *Can I still log on by using a TOTP passcode if my User ID's password expired in RACF?*

  Answer: Yes, you can still log on because your TOTP passcode replaced your RACF password. As shown in Example 7-10, a command is issued to expire the LogonID MFAUSE1 but then, the user logs on successfully by using TOTP. The PASSDATE is `00.000`, which means it expired. The logon with TOTP passcode is successful.

  Example 7-10 Expiring a RACF password

```
ALU MFAUSE1 EXPIRED
LU MFAUSE1

USER=MFAUSE1 NAME=JOHN SMITH OWNER=MVS       CREATED=13.064
  DEFAULT-GROUP=OMVSGRP  PASSDATE=00.000 PASS-INTERVAL= 60 PHRASEDATE=N/A
  ATTRIBUTES=SPECIAL OPERATIONS
  REVOKE DATE=NONE   RESUME DATE=NONE
  LAST-ACCESS=16.298/09:57:46
```

- Question: *Can I still log on by using a TOTP passcode if my user ID was REVOKED in RACF?*

  Answer: No.

A

Sample AT-TLS policy for use with MFA

The policy that is presented in this appendix is the same policy that is described in Chapter 5 of *IBM Multi-Factor Authentication for z/OS Installation and Customization*, SC27-8447.

Sample AT-TLS policy

Note: Use caution if you want to transfer this policy to your z/OS system and want to use the cut and paste function. Pay especially careful attention to the special character tilde (~), and ensure that you keep the same syntax as shown in this section. Failing to use such caution can introduce invalid pointers in your policy it might not be validated and loaded by PAGENT.

You can use the following policy as a sample. Ensure that is meets your local standards, procedures, and policies:

```
##
## AT-TLS Policy Agent Configuration file for:
##    Image: TX9
##    Stack: TCPIP
##
TTLSRule ttls_AZF
{
   LocalAddr ALL
   RemoteAddr ALL
   LocalPortRange 6789
   Direction Inbound
   Priority 255
   TTLSGroupActionRef gAct1~AZF
   TTLSEnvironmentActionRef eAct1~AZF
   TTLSConnectionActionRef cAct1~AZF
}
TTLSGroupAction gAct1~AZF
{
   TTLSEnabled On
   Trace 5
}
TTLSEnvironmentAction eAct1~AZF
{
   HandshakeRole Server
   EnvironmentUserInstance 0
   TTLSEnvironmentAdvancedParmsRef eAdv1~AZF
   TTLSKeyringParmsRef keyR1~AZF
   Trace 5
}
TTLSConnectionAction cAct1~AZF
{
   HandshakeRole Server
   TTLSCipherParmsRef cipher-AZF
   TTLSConnectionAdvancedParmsRef cAdv1~AZF
   CtraceClearText Off
   Trace 5
}
TTLSConnectionAdvancedParms cAdv1~AZF
{
   ApplicationControlled Off
   SecondaryMap Off
}
TTLSEnvironmentAdvancedParms eAdv1~AZF
```

```
{
   ApplicationControlled Off
   SSLv2 Off
   SSLv3 Off
   TLSv1 Off
   TLSv1.1 On
   TLSv1.2 On
}
TTLSKeyringParms keyR1~AZF
{
   Keyring mfa.server.KeyRing
}
TTLSCipherParms cipher-AZF
{
   V3CipherSuites TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
   V3CipherSuites TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
   V3CipherSuites TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
   V3CipherSuites TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
   V3CipherSuites TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
   V3CipherSuites TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
   V3CipherSuites TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
   V3CipherSuites TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
}
```

Related publications

The publications that are listed in this section are considered particularly suitable for a more detailed discussion of the topics that are covered in this paper.

IBM Redbooks

The IBM Redbooks publication *IBM z/OS V2R1 Communications Server TCP/IP Implementation Volume 4: Security and Policy-Based Networking*, SG24-8099, provides more information about the topics in this document.

You can search for, view, download or order this document and other Redbooks, Redpapers, Web Docs, draft, and other materials at the following website:

ibm.com/redbooks

Other publications

The following publications also are relevant as further information sources:

- *IBM Multi-Factor Authentication for z/OS Installation and Customization,* SC27-8447
- *IBM Multi-Factor Authentication for z/OS User's Guide*, SC27-8448
- *z/OS Security Server RACF Security Administrator's Guide*, SA23-2289
- *z/OS Security Server RACF Callable Services,* SA23-2293
- *z/OS Security Server RACF Macros and Interfaces*, SA23-2288
- *z/OS Security Server RACF Messages and Codes*, SA23-229100
- *z/OS Security Server RACF Command Language Reference,* SA23-2292
- *z/OS Security Server RACF Auditor's Guide,* SA23-2290
- *z/OS Security Server RACROUTE Macro Reference*, SA23-2294
- *z/OS Security Server RACF Data Areas,* GA32-0885
- *z/OS Security Server RACF General User's Guide,* SA23-2298
- *z/OS Cryptographic Services ICSF System Programmer's Guide*, SA22-7520
- *z/OS Cryptographic Services ICSF Writing PKCS #11 Applications*, SC14-7510

Online resources

For more information about available downloads, see the following website:

http://www.ibm.com/servers/eserver/zseries/zos/downloads/

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

REDP-5386-00

ISBN 0738455733

Printed in U.S.A.