Federated Physical Access Control System (PACS) Specification

Protiviti Government Services - Exostar LLC - CertiPath
U.S. General Services Administration Task Order #14
Deliverable Number 7

Table of Contents

1 Introduction *5*

1.1 Background **5**

2 Achieving Trust across Agencies *5*

2.1 Convergence **6**

2.2 Physical Asset Protection **6**

2.3 Extending Physical Protection **6**

2.4 PACS/LACS Convergence and Operational Intelligence **7**

2.5 Evaluate Options **7**

2.6 Summary of Requirements **7**

3 Typical Physical Access Architecture *7*

4 Criteria for Choosing a PACS Upgrade Approach *9*

5 Changes Imposed by FIPS 201 Compliance *10*

5.1 Summary requirements **11**

6 Validation *11*

6.1 Authentication **12**

6.2 Validation: What, When and Where **12**

6.3 Implementation Guidance from NIST **14**

6.4 Summary of requirements **15**

7 User Populations *16*

7.1 Summary Requirements **16**

8 Processing requirements *17*

8.1 Card to Reader **17**

8.2 Reader to System **18**

9 Project Management Considerations *18*

Appendix 1 The GSA PACS Demonstration System Approach *20*

Features and Benefits of F5 Approach **21**

Exostar PACS System Overview **21**

Guests **21**

Visitors **21**

Appendix 2 BridgePoint Systems Solution *24*

Appendix 3 TrustBearer Visit Request Technical Specifications *29*

Web-based digital certificate reading module for CAC and PIV/PIV-I **29**

Project Requirements **29**
Personal Identity Verification (PIV) and PIV-Interoperable (PIV-I) Smart Cards **31**
CertiPath Visit Request Website Workflow **33**
Appendix 4 CoreStreet Communications Protocols ***37***
Table of Contents **1**
1 Introduction ***3***
2 Communication Settings ***3***
3 Commands From Host to Reader/Card ***3***
4 Commands/Responses From Reader/Card to Host ***3***
5 Serial Protocol ***3***
6 Serial Commands ***6***

Acknowledgements

This document has been compiled by the Protiviti Government Services (PGS) team as part of its Task Order for demonstration of a Federated Physical Access Control (PACS) system on behalf of the U.S. General Services Administration. Special thanks are extended to Stephen Howard of CertiPath for his contributions as primary author of this document and his coordination of material and subject matter input from the project team and involved industry participants.

Certain portions of this document may be copyrighted, and are reproduced with the agreement and participation of their original authors.

1 Introduction

This specification describes the architectural level and systems level requirements for implementation of a PIV and PIV-I compliant Physical Access Control System (PACS). This document provides specific requirements for transitioning a deployed PACS to support Federated PACS system requirements, enabling interoperable credentials as described in PIV and PIV-I.

1.1 Background

An extract from the Statement of Work for the GSA PACS Demonstration system provides most of the background and objective for the demonstration system and this resulting specification:

1. **Background**
   The General Services Administration (GSA), Office of Governmentwide Policy, is responsible for government-wide coordination and oversight of Federal Identity, Credential, and Access Management (FICAM), comprised of Federal PKI, Federal Identity Credentialing (HSPD-12) and E-Authentication activities. These activities are aimed at improving Electronic government services internally, with other government partners, with business partners, and with the American citizen constituency.

   A key initiative of the FICAM is ensuring agencies have the tools to implement strong, reliable identity management programs. As agencies deploy PIV cards to their employees and contractors, the impetus to use the capabilities of the credentials to gain access to Federal facilities increases. It is in the best interests of the United States Government to research the ability to deploy and use physical access control systems that are responsive to PIV cards and PIV-Interoperable (PIV-I) cards.

2. **Objective**
   The objective of this Statement of Work (SOW) is to demonstrate physical access control interoperability leveraging PIV and "PIV Interoperable" identity cards across multiple PKI bridge CAs.

This specification provides the architectural guidance, requirements, and technical specifications for implementation of a Federated PACS system leveraging PKI enabled PIV and PIV-I credentials. This specification complies with the following requirements:

> "Subcontractor will create a vendor agnostic, generic systems requirement and architecture document that describes the system in sufficient detail to allow interested vendors to create and subsequently market a conformant PACS implementation."

Appendices are used to describe details of vendor specific information from the completed project.

2 Achieving Trust across Agencies

A major goal of HSPD-12 is to achieve trusted interoperability throughout the federal government. This trust is achieved by, among other things, having a highly secure identity card that supports strong authentication mechanisms. The PIV card has several built in security features, including:

- Cryptographic signing of integrity sensitive data elements such as certificates, fingerprints and facial images
- An on-card FIPS 140-2 certified cryptographic module
- FIPS 140-2 Security Level 3 protection of private keys stored on-card
- Keys are generated on-card and the private keys cannot be exported
- All cryptographic operations using the private keys are performed on-card
- Access to card commands involving the PIV private key require a PIN
- Cardholder PIN is matched on-card

These features can strongly enhance the overall security of a physical access control system if used properly. Ensuring the proper use and validation of the PIV credentials is at the heart of the process of upgrading an existing PACS to make it consistent with the intent of HSPD-12. This intent drives us to requirements where entering a facility should be just as secure as logging onto a PC.

Security professionals are finding themselves hard-pressed to secure both physical *and* logical assets (as well as comply with stringent government mandates). So why are so many organizations behind the curve when it comes to managing physical and logical access?

There are dozens of excuses: "It's too expensive." "Securing electronic data is a greater concern." "We wouldn't even know where to start." But the reality is that a unified approach to physical and logical access may actually save money – not to mention time. And while cyber attacks are a mounting threat, physical/logical access control loopholes can be Achilles' heels to sophisticated hackers who know how to exploit them.

High assurance identity credentials (PIV and PIV-I) allow organizations to manage access to secure areas *and* systems. The following five concepts drive architectural decision making at an organizational level. They help to ensure that your organization does not fall victim to common access control risks.

2.1 Convergence

Do not treat physical and logical access control separately. Both are about controlling access to a resource – they share the same security goal. Whether that resource is a sensitive room or a sensitive piece of data, access rules will be defined in the same manner. Similarly, the same identity information about the requestor can be required to evaluate access requests.

CISOs at many organizations struggle to justify the cost of high assurance identity credentials for use in their IT systems. CSOs have struggled with this same cost vs. benefit problem for high assurance PACS capabilities, such as biometric readers.

Today, enterprises creating successful business cases look at physical and logical access as the same problem that can leverage the same solution: PKI and PIV/PIV-I credentials. Convergence can save money and improve security.

2.2 Physical Asset Protection

Do not allow physical security to lag behind logical security. Criminals attack the weakest link, and at many organizations today, that is likely to be the front door – literally.

There have been many cases where data thefts occurred not online, but rather through lax physical security of the servers themselves. Remember Willie Sutton's famous quote about why he robbed banks: "Because that's where the money is."

2.3 Extending Physical Protection

If you have embraced the benefits of PIV and PKI federation for your web portal or cloud applications, don't stop at your front door. Identity Federation using PIV is commonly accepted within federal agencies as the most effective way to gain assurance of the identity of persons external to your organization. PIV-I credentials strengthen this posture using Medium-Hardware assurance levels for organizations outside the federal government.

In other words, I accept my partner's own agency issued credential for access into my applications. I receive the most up to date identity information about my partner, verification of their employment status and I avoid having to provision and maintain credentials for these external users. The most mature identity federation organizations though *still* issue me a temporary badge when I show up in their building's lobby, despite being able to accept agency issued credentials at their website.

If you consider that visitors to your offices are there for a business purpose that is related to whom they work for and in what role, then this is critical to verify at the time access is requested. If their employment relationship no longer exists (e.g. they were fired), this would be critical to know before allowing access to the building. But this vetting is rare, because it's socially awkward and tedious to do manually. Interoperable credentials and a trust framework that backs them can allow an organization to leverage their partners' credentials for PACS and LACS simultaneously.

2.4 PACS/LACS Convergence and Operational Intelligence

PACS/LACS convergence is about more than cost savings or increased assurance of identities; it is about better operational intelligence.

CISOs and CSOs have not traditionally talked to one another (often the case even when it is the same person in both roles) given the silo-like nature of these areas. What opportunities are missed as a result? If a user logs in from home on their VPN, and the same person has just "badged in" at the office, isn't that a problem?

Even PACS talking to PACS in the same organization is unusual. What if a user badges into their home office in the morning and the branch office across the country an hour later? Attackers are looking for blind spots and the "PACS/LACS barrier" represents tempting low-hanging fruit.

2.5 Evaluate Options

Look to your industry or immediate customer base to determine if there are already others that have implemented converged PACS/LACS solutions with external partners in mind.

In the United States, the Federal Government's PIV and PIV-I are the dominant high assurance credentials intended to be used for both PACS and LACS. Everything from desktop login to email can already take advantage of PIV or PIV-I based credential but recently PACS vendors have released systems compatible with these credentials as well. Converged PACS/LACS solutions are a now mainstream use case. This specification describes how to achieve these goals using the GSA Federated PACS Demonstration system architecture.

2.6 Summary of Requirements

Do not treat physical and logical access control separately.

Do not allow physical security to lag behind logical security.

If you have embraced the benefits of PIV and PKI federation for your web portal or cloud applications, don't stop at your front door. Cyber security can *include* PACS.

PACS/LACS convergence is about more than costs savings or increased assurance of identities; it is about better operational intelligence. Leverage PKI and Federation to the benefit of LACS and PACS in a cohesive security model.

3 Typical Physical Access Architecture

A typical PACS architecture will look similar to that shown in Figure 1 below. While different PACS vendors may name their components differently, the essential functionality of all systems is the same[1]. Before launching into a discussion of changes required to enable a FIPS 201 PACS solution, it is important to understand the current state of the industry.

1 Panels are sometimes referred to as Intelligent System Controllers or PACS Field Controllers. Door controllers are sometimes referred to as Reader Interface Modules

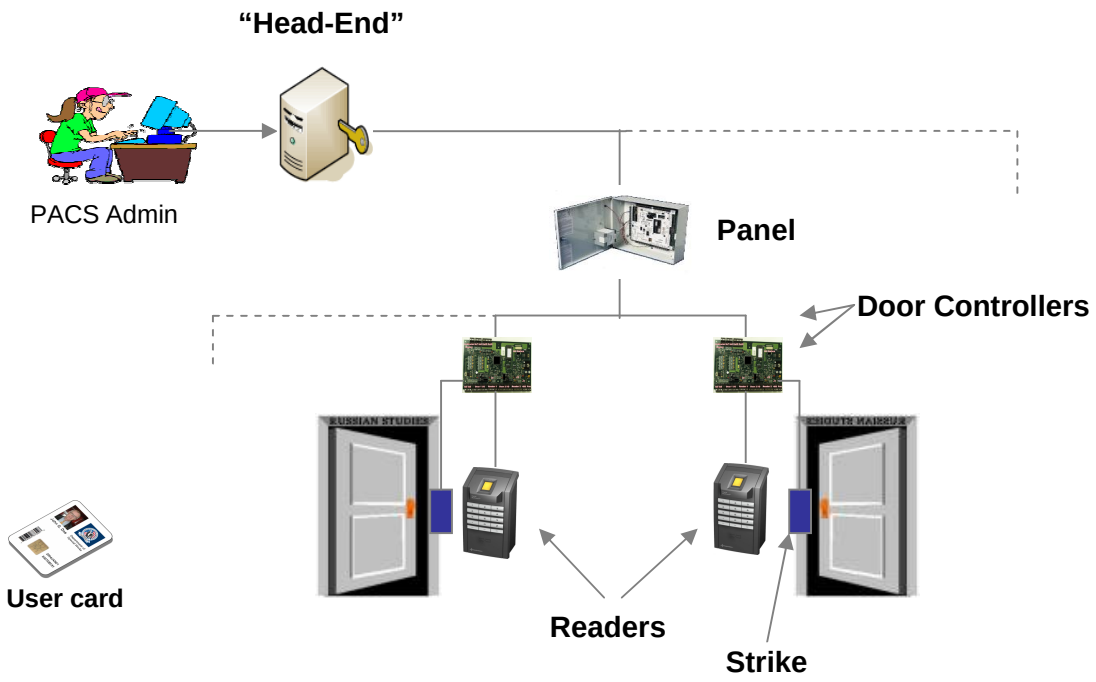


Figure 1: *Typical Physical Access Control System (PACS)*

The basic components of a PACS are the head-end server, panels, door controllers, readers, lock or strike mechanisms and the user identity cards. Each of these components is described below.

The **head-end** server is where the system is managed. This includes the following functionality:

- Enroll[2] users (name, …)
- Assignment of one or more unique badge IDs to the user
- Maintain status of enrolled users (active, inactive)
- Maintain user access privileges (often called access levels)
- Maintain door attributes (e.g. lock ID, type, software version, etc.)
- Maintain door policies (level required for access, time based requirements)
- Collects, manages and reports access events (e.g., who entered, when, etc.)

The head-end server is the "brains" of the system and must therefore be located in a secure location.

The **control panel** is the access control "decision maker". Its role is to:

- Maintain and enforce door policy by matching user access level with door access level
- Stores access control data locally so that it can continue to operate when connectivity to head-end is lost
- Executes alarm and sensor logic
- Contains battery backup so that it can continue to operate during loss of power

A given PACS may have several panels depending upon the size of the system.

The **door controllers** provide the mechanical interface to execute the access control decisions. This involves:

- Driving the door strike
- Providing audio signals to indicate door latch is open
- Providing LED signals to indicate access control decisions (access granted or denied)
- Providing on/off signals for video recorders, door ajar alert, manual "buzz in", etc.

Depending upon the manufacturer, a door controller can control one, two or several doors.

2 In this document we use the term "enroll" in the sense of provisioning users into a PACS head-end.

The **readers** are familiar to all of us. Their primary purpose is to provide an electrical interface to the User ID card. Their functions include:

- Providing an access request presentation point to user
- Reading the claimed identity data from card
- Accepting authentication data (e.g., PIN, biometric)
- Communicating identity and authentication data to panel via the door controller
- Providing LED and audio on/off feedback to cardholder

Readers come in a variety of types: contactless, contact, with PIN pads, with biometric sensors and in various combinations.

Common **lock/latch mechanisms** in use are:

- Door strike – typically wired back to door controller where the latch is controlled by door controller (as shown in Figure 1)
- Lock – typically not wired back to door controller. In this case the reader is built into lock which contains an access control list (ACL) and access policies specific to the lock.

Various **identity card** types are currently in use by physical access control systems. Examples include:

- Magnetic Stripe
- Proximity (e.g., HID PROX)
- Smart Card (e.g., PIV, iClass, DESFire, MIFARE). Smart cards are further characterized by type of interface (contact, contactless or dual) and size of processor, and memory.

Perhaps the most defining characteristic of existing systems is that user identity cards are issued locally by each PACS. This means that the choice of card type, format of the data on the card and assignment of badge IDs are all site dependent. In addition, there are very few standards associated with existing physical access control systems. Readers typically use the Wiegand protocol to talk to the door controllers but most other components and protocols are proprietary, including the identity cards. The result is that there is virtually no interoperability between separate PACS, even those from the same vendor.

4 Criteria for Choosing a PACS Upgrade Approach

The focus of this paper is on upgrading existing PACS deployments as opposed to deploying a new PACS. Regardless of the approach taken there are certain goals that apply. In general, a suitable approach to PIV enabling an existing PACS should meet the following criteria:

- **Maximize reuse** – the approach should not be based on a rip and replace strategy as this is not required. Maximizing reuse is key to minimizing cost.
- **Minimize custom modifications** – custom modifications are typically expensive and difficult to maintain or replace and should be avoided. A commercial off the shelf approach should be used wherever possible.
- **Support multiple PACS** – many organizations are not standardized on a single PACS make or model for their entire enterprise. A PIV enabling solution that is not tied to a specific make or model will make future upgrades of the PACS components much easier.
- **Support multiple authentication mechanisms** – NIST Special Publication 800-116 [SP 800-116] identifies four authentication mechanisms suitable for controlling access to "controlled", "limited" and "exclusion" areas[3]. The PIV enabling solution should support all of these mechanisms and provide the capability to dynamically switch between them in response to changes in threat level.

3 See also Appendix B for a description of these controlled areas.

- **Support PIV-I** – PIV Interoperable[4] cards are used by federal contractors who are included in the HSPD-12 mandate and may need access to a controlled facility. This capability will enable PIV-I visitors as well as temporary PIV-I cardholding employees to use the access control system.
- **Improve security** – HSPD-12 is all about improving security in both physical and logical access control. To be effective the solution must securely execute the recommended authentication mechanisms.

5 Changes Imposed by FIPS 201 Compliance

The introduction of the PIV card represents a major step forward in standardization of access control within the federal government. There is now one standard identity card that is centrally issued and is recognizable and trustable by all government agencies. While using the PIV card in existing PACS will require some changes it will not necessitate a complete replacement of the PACS components. *Figure 2* below shows where these changes affect the system.

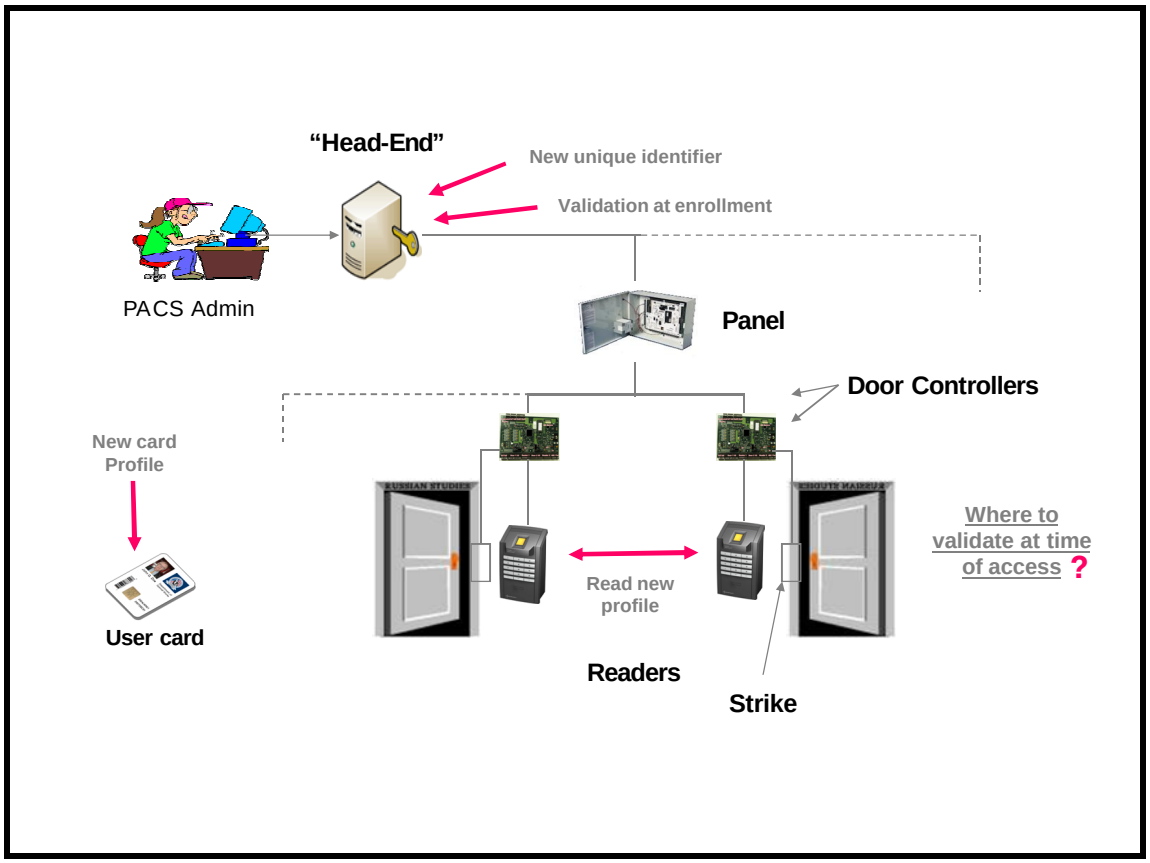


Figure 2 - FIPS 201 Changes to PACS

Upgrading an existing PACS to enable it to properly use a PIV card as the user identity card requires a few small but significant changes:

1. The PIV card is as an ISO 14443 type smart card with a contactless interface that operates at 13.56 MHz. In addition some authentication mechanisms require using the contact interface. The most common identity cards in use today are contactless proximity cards which operate at 125 kHz. This

4 PIV-I cards are defined as "an identity card that meets the PIV technical specifications to work with PIV infrastructure elements such as card readers , and is issued in a manner that allows Federal government relying parties to trust the card." See reference [PIV-I].

incompatibility in communication protocol and the need in some cases to support the contact interface will require replacement of the readers.

2. The PIV card employs a new profile for representing the data on the card. The system must therefore add functionality to read and interpret this new profile.
3. Each PIV card contains a unique identifier called a FASC-N[5]. The unique identifier on PIV-I cards is the GUID[6]. The GUID contains a UUID value in accord with RFC 4122. New functionality must be added to extract the unique identifiers from the card data and use them in the access control decision process.
4. To ensure secure use of the PIV card some level of authentication and validation must be performed as part of the enrollment process and at the time-of-access. This is new functionality that must be added to the system.

5.1 Summary requirements

1. Support for multiple authentication modes
2. Bidirectional communication with the reader is mandatory
3. Enterprise security policy and incident response policy must be implementable across PACS
4. E.g. the facility must be put into lockdown during an incident
5. Guests cannot be handled out-of-band
6. They are a normal condition in most buildings and not unusual
7. Visitors that carry credentials of similar, provable quality must be leveraged
8. Leverage the HR function's proofing of each employer that supports that credential
9. Do not issue "native" badges to compensate in facilities with heterogeneous populations
10. All processing of authentication information must occur on the "safe side" of a door

Shoulds:

1. Should be accredited to a agreed upon standard
    1. SSL headlines in 2009 have something to teach us
2. Provide the same security functionality found in LACS
    1. e.g. temporal policies to effect reader authentication modes
    2. e.g. mutual registration of card and reader
    3. Lots of feedback to the user at the reader (status bar)
3. Geographical and temporal velocity checking:
    1. People who entered a facility in Washington DC should not be seen an hour later in the New York office nor 3 hours later in the San Francisco
    2. Take advantage of PACS/LACS convergence
        - People entering the front door should not also be logged in remotely
4. Should quickly be adopted in other resource sensitive applications outside of Government and B2G
    1. Guardian/child pairing leaving day care (anti-passback)
        - Great use case for Iris

6 Validation

Of the four PACS changes identified above, adding PKI based validation for the PIV credential is the most complex. The term "validation", as used in this document, is defined to be the process of authentication (i.e., proving your claimed identity) and revocation checking of the presented credential.

---

5 FASC-N stands for Federal Agency Smart Credential Number.
6 GUID stands for Global Unique IDentifier. See reference [PIV-I] for additional information.

6.1 Authentication

Authentication is the process of proving one's claimed identity. There are three basic threats associated with authenticating a user claiming to be the person identified on the PIV card:

1. Counterfeit identifiers – a genuine PIV credential is issued by a trusted authority. At the point and time of access it is imperative to know that the presented identifier is genuine and has not been forged by someone seeking unauthorized access. This threat is mitigated through the use of digital signatures on each of the credential's data objects (e.g., certificates, fingerprint template, facial image). Validation of these signatures ensures the data were signed by the trusted authority.
2. Cloned or copied identifiers – trusting a PIV credential certificate requires knowing that it is not a copy of a legitimate user's certificate. This threat is mitigated by executing a PKI private key challenge to ensure the certificate (through its public key) is bound to the private key embedded in the PIV card.
3. Lost or stolen identifiers – trusting the identifier requires knowing that it represents the person presenting it. This threat is mitigated by verifying the binding of the card holder to the card by requiring either a PIN, biometric or both as part of the validation process.

In general the decision to grant or deny access is not based on authentication alone. The person requesting access must also be authorized to do so and their identity credential must be checked to ensure it has not been revoked by the issuing authority.

6.2 Validation: What, When and Where

Validation of the PIV card would typically include some or all of the following:

1. Path discovery – the process of discovering a path from the PIV certificate to an embedded trust anchor.
2. Path signature verification – establishing that every certificate in the path is genuine and not counterfeit.
3. Data object signature verification – establishing that every signed data object on the card was signed by a trusted issuer (e.g., certificates, fingerprint template, facial image template) to ensure they are genuine and not counterfeits.
4. Cross checking data object identifiers – all signed data objects on the PIV card have an identifying number (FASC-N) unique to that card. Checking that each data object contains the same FASC-N (or GUID) ensures they all belong to the same credential.
5. Various PKI conformity and freshness checking (key usage, expiration dates, etc.).
6. PIN check – to ensure the card holder is bound to the credential to mitigate the threat of lost or "shared" cards.
7. Private key challenge – to ensure the certificate is bound to the token to which it was issued and has not been copied or cloned.
8. Biometric check – to ensure the card holder is the same person that was issued the PIV card. This mitigates the threat of "shared" cards and disclosure of the card's PIN.
9. Periodic checking of the revocation status of the PIV Authentication Certificate
10. Periodic revalidating the full path – to ensure all of the certificates in the path remain valid and have not been revoked.

Validation during enrollment should include all of these checks to ensure at the highest level possible that all enrollees are in fact who they claim to be. This would typically be done as a function at or in conjunction with the PACS head-end.

Validation at the time-of-access will involve a subset of these checks depending upon the assurance level required and authentication mechanism chosen for the specific access point being addressed[7]. The question of where this validation should be done however is more challenging. Possible options include placing this functionality in:

- Head-end
- Readers
- Panel/Door Controllers

7 See NIST references [SP 800-116] and [SP 800-73-2] for descriptions of the various authentication mechanisms and associated validation procedures.

- Separate module

Performing time-of-access validation at the head-end would seem to be a logical choice since it could take advantage of the full PKI validation functionality used for enrollment. However this approach would require adding new wiring to support two way communications between the readers and the head-end to facilitate signature checking on the credential data elements (CHUID, certificates, biometric templates), execution of a private key challenge and execution of a biometric match. In most cases this would be prohibitively expensive. In addition, this approach would fail to operate properly during a loss of power or reader to head-end communication.

Alternately the replacement readers, something that may need to be done regardless of the approach chosen, could include the validation functionality. The new readers would require more powerful (and costly) processors to perform the cryptographic processes associated with PKI validation. This approach also requires two way communications with external networks or the head-end in order to receive periodic downloads of certificate status data, trust anchors for signature verification and to service path discovery requests for any visitors with PIV or PIV-I cards. In addition there are two security issues associated with this approach:

- Security related processing on the unsecured side of the PACS boundary
- PACS network connection available on the unsecured side of the PACS boundary

Putting the time-of-access validation into the panel or door controller components is a more attractive approach in that it addresses most of the deficiencies and security issues associated with the approaches discussed above. Specifically,

- Security related processing is on the secured side of the PACS boundary
- PACS network connection is on the secured side of the PACS boundary
- There is potentially much less rewiring to be done
- The system would continue to operate properly as these components typically have battery backup for lost power and could store the required validation information locally.

The drawback with placing the PKI validation functionality into a panel or door controller is that these components would have to be replaced rather than upgraded. Replacement is necessary because upgrading existing boards would require some or all of the following changes:

- Addition of a more powerful processor to perform the cryptographic PKI operations in a timely manner
- Addition of external serial ports for two way communications with the readers
- Addition of Ethernet port for communication with head-end and/or external networks for periodic retrieval of status information

Note that any such upgrade would also have to be done separately for all makes and models of these components which would not be cost effective.

The following table summarizes the suitability of each of these approaches for validation at the time-of-access.

Table 1: ***Suitability of approach to validation at the time-of-access***

Time of access validation approach —>	Head End	Reader	Panel	Plug-in
Works with existing wiring in most cases	No	No	Yes	Yes
Operates properly during power loss	No	Yes*	Yes	Yes
Operates properly during network conductivity loss	No	Yes*	Yes	Yes
Secure processing within PACS boundary	Yes	No	Yes	Yes
Network connection within PACS boundary	Yes	No	Yes	Yes

* Assumes battery backup and caching of validation data at the reader

An alternative to upgrading or replacing all the panels and/or door controllers is to augment the existing system functionality with the addition of a new "plug-in" module. To be successful this new module must work with existing PACS panels and door controllers as they are, without requiring any changes. This approach should then be the most cost effective way to PIV-enable an existing PACS.

6.3 Implementation Guidance from NIST

A prerequisite for planning and implementing a solution to PIV enable a PACS is to determine just how much security is required and where. NIST's Special Publication 800-116 provides excellent guidance in answering this question. *Figure* below is taken from this document and shows the recommended segmentation of access control points in terms of security based risks.

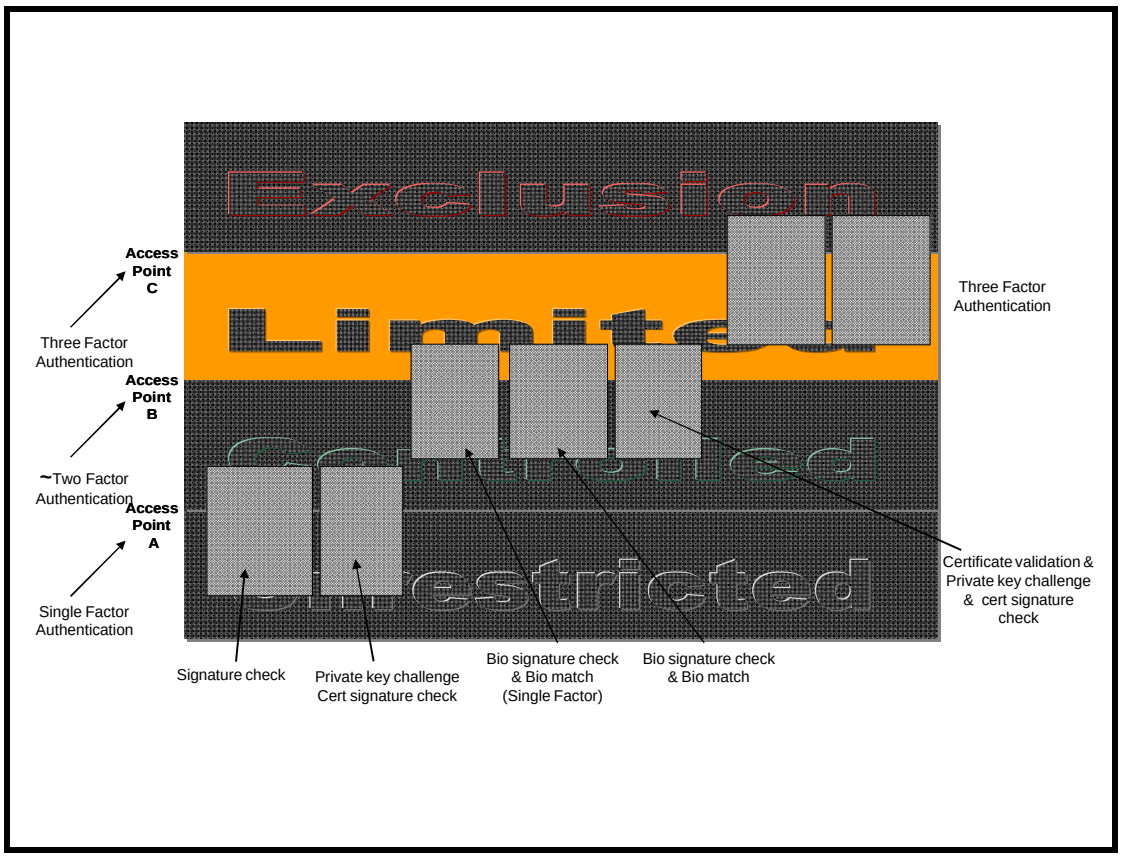


Figure 3 – SP 800-116 area definitions

In this diagram the Unrestricted area is consider public with no restrictions as to who has access. Access to the Controlled area is restricted to those who can prove affiliation. For example, possession of an Agency's badge could be sufficient to gain access at an outer perimeter of a facility. Access to the Limited area is restricted to members of a group who are fulfilling a specific role. Finally, access to the Exclusion area is restricted by individual authorization, analogous to the "need-to-know" requirement in the classified world.

SP 800-116 defines and describes the authentication methods shown in Figure 3. The PIV card is designed to support each of these methods which are intended to provide different levels of assurance of the identity of the user. Which authentication method should be used depends upon the security requirements at the point of access.

6.4 Summary of requirements

Path discovery and validation (PDVAL) should be done at the time of enrollment for all certificates. Certificate status shall be periodically rechecked for all enrolled certificates within the 18 hour mandatory time for revocation status checking, per the FPKI Common Policy. Full path validation should also be periodically repeated on a weekly or monthly basis.

The following requirements are extracted from "NIST Recommendation for X.509 Path Validation", May 3, 2004[8]. Specifically sections 3, 4 and 5. These requirements must be met to claim full PDVAL functionality by a PACS system.

The PACS System shall have an Enterprise Path Validation Module (PVM) component that processes X.509 certification paths composed of X.509 v3 certificates and X.509 v2 CRLs. The PVM component MUST support the following features:

---

8 PDVAL Steps are defined in
http://csrc.nist.gov/groups/ST/crypto_apps_infra/documents/NIST_Recommendation_for_X509_PVMs.pdf

- Name chaining;
- Signature chaining;
- Certificate validity;
- Key usage, basic constraints, and certificate policies certificate extensions;
- Full CRLs; and
- CRLs segmented on names.

An Enterprise PVM may support additional path processing functionality, including support for OCSP and SCVP.

In addition to meeting the requirements of an Enterprise PVM, a Bridge-enabled Path Validation Module must be capable of processing the types of certificate extensions that are typically found in CA certificates that cross enterprise boundaries. The three packages of requirements for Bridge-enabled PVMs

- Name Constraints,
- Policy Mapping,
- and anyPolicy.

A PVM that meets all of these requirements is considered to be Bridge enabled.

6.4.1 Basic Path Processing

The PVM shall verify that digital signatures and public keys in the certification path chain in accordance with RFC 3280, using RSA PKCS#1 with SHA-1 in accordance with RFC 3279. (That is, the PVM shall verify that the RSA with SHA-1 signature on each certificate in the path verifies using the RSA public key in the preceding certificate, and the RSA with SHA-1 signature on the first certificate in the path verifies using a trust anchor's RSA public key.)

11.

The PVM shall verify that issuer and subject names in certification paths chain in accordance with RFC 3280. (That is, the PVM shall verify that the issuer of each certificate in the path was the subject of the preceding certificate, and the issuer of the first certificate in the path is the name associated with the trust anchor public key).

7 User Populations

Identify your user population. At a minimum, there are three specific groups for consideration: guests, visitors, and regular access.

- **Guests** are individuals who do not bring a credential from a source that can be trusted by the local system. These individuals will be logged and issued Guest Badges consistent with standing or traditional escort rules.
- **Visitors** are individuals who bring a credential that was issued by their own company or another source that can be trusted by the local system. Trust is established with CAC, PIV or PIV-I through use of Medium Hardware certificates (MHW). Upon proper registration of these credentials to the local PACS, they use their own credential for access to the facility.
- **Regular Access** individuals have a credential issued by the local authority and have regular access to the facility.

7.1 Summary Requirements

Define user populations: Guests, Visitors, Regular Access.

Define access points for each population.

Define temporal access rules for each population.

Define challenge and verification program for each population.

Choose authentication approach for each population *and* each zone/point of access in accord with NIST SP800-116.

8 Processing requirements

This section specifically enhances the guidance found in TIG SCEPACS v2.3. That specification and the FIPS 201 standards bring the card to the reader, but do not define requirements for operations within the relying party system. These requirements are derived from the operational experience of the GSA Federated PACS Demonstration.

8.1 Card to Reader

Prepare a card discovery tree that describes the Card Types supported and the Authentication Modes required for the system. Prepare a mapping of Card Type to the supported Authentication Modes available within that card type. A summary of card types and modes supported by the PACS demonstration are:

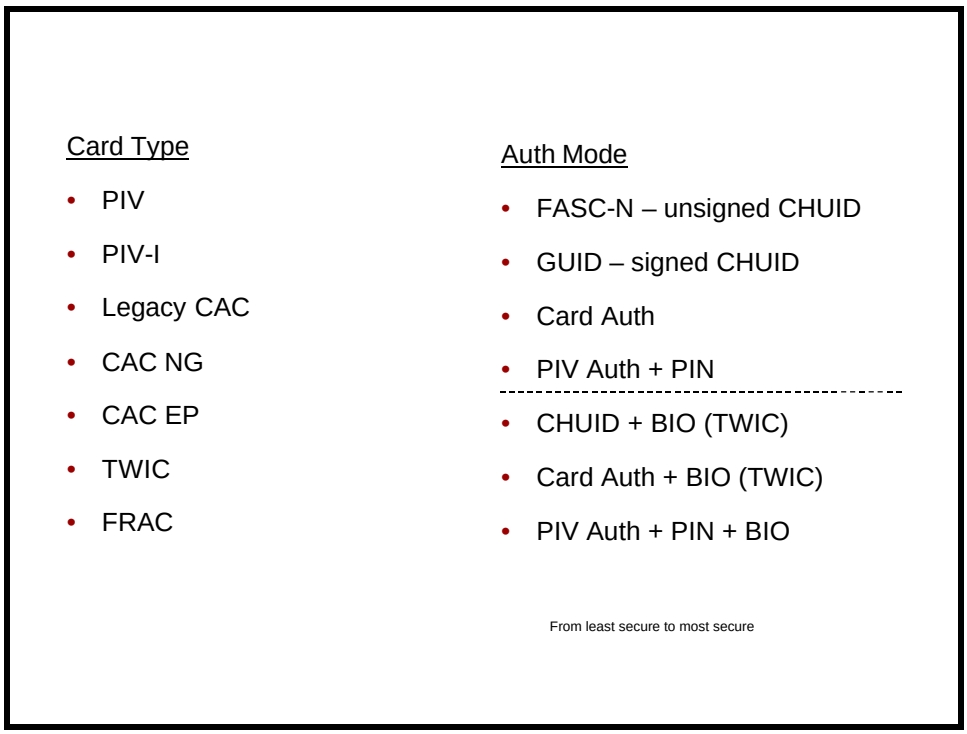


Figure 1 - Card Types and Authentication Modes

The following acronyms are defined from

Figure 1:

- PIV – Personal Identity Verification credential as defined by NIST SP800-73
- PIV-I – PIV Interoperable credential
- Legacy CAC – Contact only Common Access Card with v1 and v2 applets
- CAC NG – CAC Next Generation with transitional PIV applet
- CAC EP – CAC End Point with end point PIV applet
- TWIC – Transportation Worker Identification Credential
- FRAC – First Responder Authentication Credential

The card discovery tree may look as follows:

1. PIV (Includes CAC EP, PIV-I, FRAC and TWIC)
2. CAC NG
3. CAC

This model has the following issue:

- If you are using TWIC's contactless biometric application, you may have to search for that first in your discovery tree. PIV masks the fact that you are seeing an actual TWIC credential.

If this issue impacts your system, the discovery tree may need to be modified to become:

1. TWIC
2. PIV (Includes CAC EP, PIV-I and FRAC)
3. CAC NG
4. CAC

If your primary user population is any one of these card types, you may seek to alter the discovery tree by moving a particular dominant card type earlier (e.g., search for CAC first for DoD installations). This is not recommended. The migration to PIV will become dominant over the next several years.

8.2 Reader to System

When designing an access control system, you do not want security relevant decisions being made on the attack side of the door. An attack vector is to scan for CHUIDs. Once these are identified, simply remove the reader from the wall. Attach a PC to the reader's cable and send the appropriate CHUID or FASC-N up to the door controller. This signals the door controller/panel to open the door, as the assumption is that the reader already performed the security relevant processing of PKI, PIN and/or biometrics. This is essentially a yes machine attack scenario.

The following requirements apply to Federated PACS systems.

8.2.1 Attack side processing requirements

No security relevant decisions are made by system components that do not belong to the cardholder's credential when they are on the attack side of the door. This specifically applies to the door reader.

The door reader shall be a "transparent" reader that cooperates with the system to present information to the secure side that enables security relevant decisions.

If these requirements are not met, there may be compensating controls applied, such as tamper switches and FIPS 140-2 certified cryptographic processing within the reader itself.

8.2.2 Secure side processing requirements

The secure side shall use an open, transparent protocol to communicate with the reader on the attack side. See Appendix 4 for an example of an open protocol provided by CoreStreet.

Secure side processing shall include:

- Challenge/response management
- PKI path discovery and validation
- Credential identifier processing
- Authorization decisions

9 Project Management Considerations

As Federated PACS solutions incorporate aspects of both IT and physical security, program and project management practices should be formalized.

In particular, key risks should be considered early in project planning – and managed in a manner similar to an IT or ERP system rollout:

- Ensure that the project scope is well-established in a **documented concept of operations**.

- Ensure adequate **stakeholder support** from all parts of the organization.

- Ensure a positive user community experience with **formal change management and communications plans**.

- **Phased migrations and production pilots** are critical to ensuring a successful large-scale rollout.

- Ensure that all IT and physical security considerations are defined, with **frequent and increasingly detailed design reviews** throughout the project.

- Work closely with system integrators and local authorities (e.g. city or county fire marshals and local law enforcement agencies) to **ensure adequate lead times for third-party design and permit reviews**. The timeline for approvals may vary significantly, depending on particular local jurisdictions and building/fire codes.

- Particular care should be exercised during retrofits of existing, occupied spaces, as **continuity must be maintained in both access and security**.

Appendix 1 The GSA PACS Demonstration System Approach

The demonstration system's approach leverages Veridt and CoreStreet technologies to PIV enable an existing PACS as shown in *Figure 2*. In this approach the PIV enabling functionality is added by augmenting the existing door controller and panel functionality. It requires two changes: replacing existing card readers with Veridt PIV enabled readers and inserting a CoreStreet F5 module between the reader and the door controller. The F5 module contains all the PKI validation functions executed at the time-of-access and it operates within the secure environment. It is installed next to the door controller.

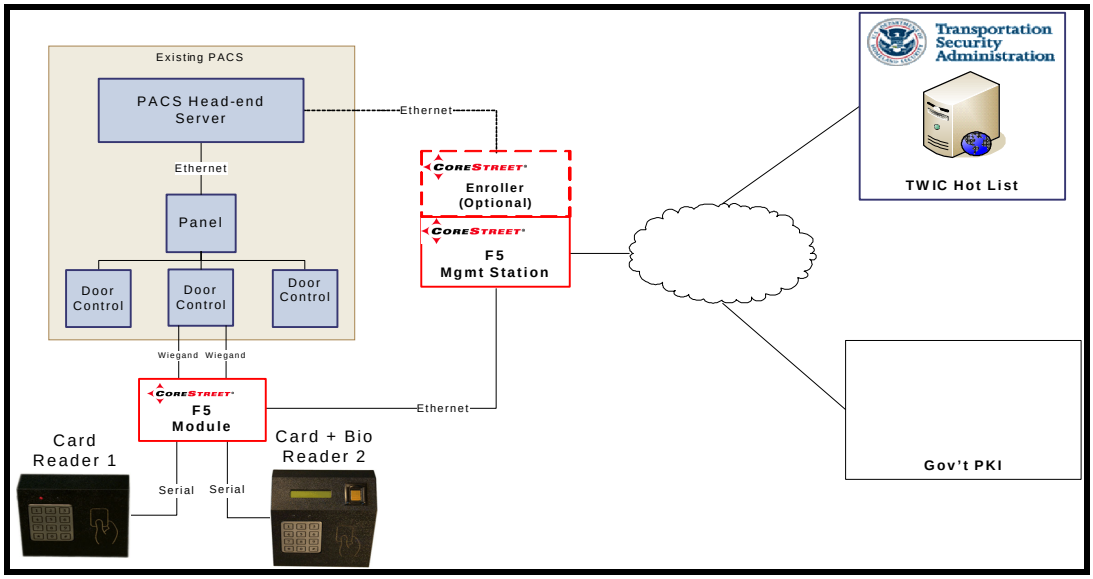


Figure 2 - PIV Enabled PACS using the Veridt Reader & CoreStreet F5 Panel Augmentation Approach

Inserting the CoreStreet F5 module requires no modification or replacement of any non-reader component in an existing PACS. It provides all the PKI validation functionality required by FIPS 201 in compliance with HSPD-12. F5 modules can be installed between any existing PACS panel or door controller and a supported reader: contact card-only, contactless card-only, contact and contactless card-only, card + PIN, card + bio or card + PIN + bio. Readers are selected based on assurance level requirements. The F5-reader combination supports CHUID, CAK, PKI, and BIO authentication methods as described in SP 800-116. Each F5 module can support one or two readers.

F5 modules are managed by an F5 Management Station (F5MS) that provides centralized control of assurance level settings and distribution of dynamic validation data such as credential revocations and trusted issuers. The F5MS also controls pushing firmware updates to all F5 modules.

The F5 validates cards according to its assurance level setting, constructs the badge ID from data on the card and then passes the badge ID to the PACS panel for an access decision. The PACS Head-end maintains the user access authorizations as is currently done. For invalid cards, the F5 can be configured to send a preset badge ID to the PACS panel and/or close an output relay. Cardholder data is captured automatically the first time a card is presented for access and then stored at the F5MS. This feature allows traditional enrollment of cardholders using existing PACS enrollment functionality, integration with an identity management system (IDMS) or use of a third party enrollment package such as visitor software or the CoreStreet Enroller.

Features and Benefits of F5 Approach

CoreStreet's F5 approach enables an agency to PIV enable their existing PACS system to fully leverage federation and PKI in a cost effective and secure manner that meets all of the previously defined criteria:

- **Maximizes reuse** – the F5 solution minimizes cost by augmenting the capability of existing panels and door controllers and requires no changes to the existing system other than adding PIV compatible readers.
- **Minimizes custom modifications** – the F5 solution does not require any custom modifications to existing PACS components. Future upgrades to the existing PACS can be done without requiring any custom modifications.
- **Supports multiple PACS** – the F5 solution is PACS make and model independent. The optional Enroller component can be integrated with various PACS head-ends.
- **Supports multiple authentication mechanisms** – the F5 solution provides dynamically configurable support for all authentication mechanisms defined in SP 800-116 (CHUID, CAK, PKI, BIO and combinations).
- **Supports PIV-I** – the F5 solution supports a variety of identity credentials in use today, including PIV, PIV-I, TWIC, FRAC and CAC (legacy, NG, EP). All four TWIC authentication modes as defined in the TWIC reader specification are supported.
- **Improves security** – the F5 solution provides a complete PKI validation approach to support strong authentication of the card holder. This includes configurable periodic status checking via OCSP, CRL or TWIC hot list and validation of contractor and visitor identities via certificate path discovery and validation through the Federal Bridge.

Exostar PACS System Overview

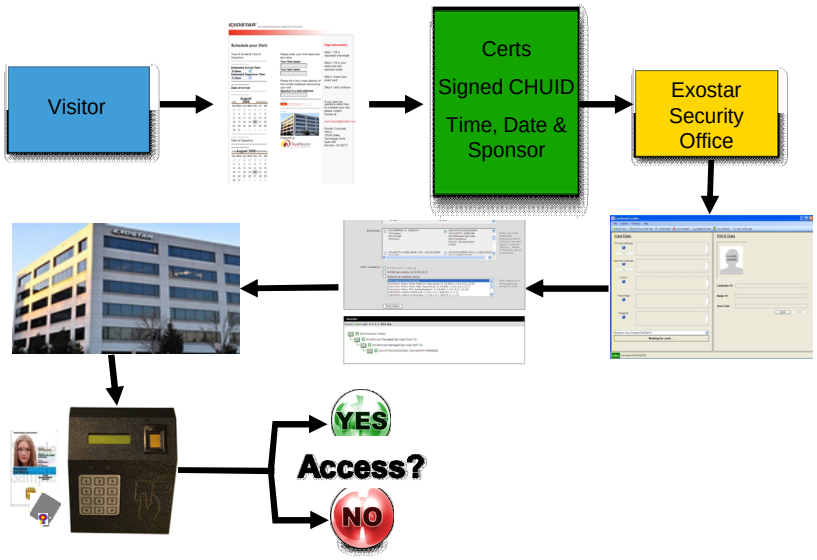
Guests

Guests are treated in the manner commonly used today where they must sign in to a paper log and receive a locally issued guest badge. The guest badge must be visually distinct to ensure proper management of individuals with guest status.

Visitors

The GSA Federated PACS demonstration uses the following process for visitors.

Process Overview



Pre-registration of certificates and IDs

In advance of a visit, a cardholder accesses a website (https://visitrequest.exostar.com) with his or her CAC or PIV/PIV-I smart card and makes a visit request. This request provides the following information:

- Visitor's name
- Sponsor's email address
- Date and time of beginning of visit
- Date and time of end of visit (date enables multi-day visitors)

Upon hitting the continue button, the visitor's CAC, PIV or PIV-I credential is read for free read information including:

- PIV Authentication certificate
- Digital Signature certificate (if available)
- Key Management Key certificate (if available)
- Card Authentication Key certificate (if available)
- CHUID (if available)

This information is routed to the physical security office for the following steps:

- Confirmation of sponsorship
- Approval of visit request
- Grant of access into the local PACS
- Pre-registration of CHUID and digital certificates into the system

Prior to the cardholder's visit, the system verifies that the cardholder's certificates were issued by a valid source and that the card has not been revoked. This proactive certificate processing saves valuable seconds of time when the cardholder presents his or her CAC or PIV card at the door reader.

Visitor arrives and access requested/granted

The requirement is for the individual to use their existing credential that was registered using the visit request process at the intended point of access for visitors. This is typically a lobby area that is attended.

The door reader will perform a challenge-response authentication with the card in real-time, but not certificate path discovery & validation (PDVAL) checking. The door reader does perform signature validation of the certificates, linking the existing certificate PDVAL activities to the proper certificate.

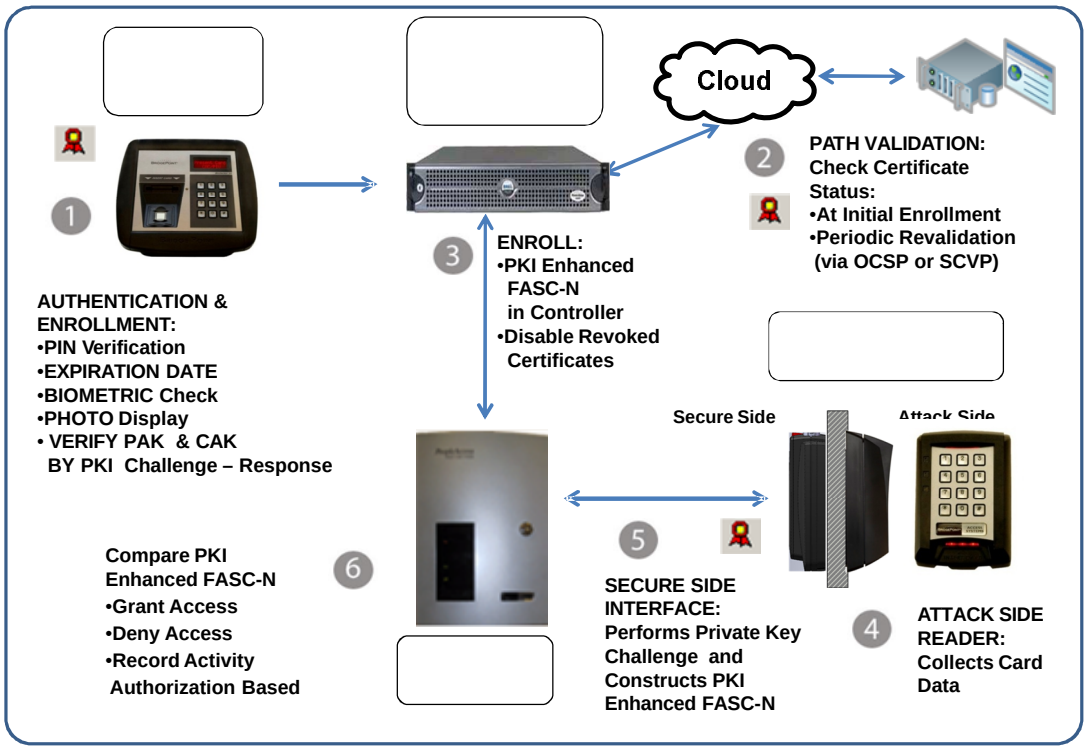
Appendix 2 BridgePoint Systems Solution

HOW TO IMPLEMENT BRIDGEPOINT PACS WITH PKI

Follow Guidance in NIST 800-116 to establish a Viable Migration Strategy	
Establish Security Level to be met by Area to be Secured	Determine Number of 1F, 2F and 3F Readers Determine use of contact, contactless interface and use of private key challenge response (PAK and CAK) Are any Readers subject to attack on the attack side of the door (sending false PKI or BIO match information) Are Exit Readers required to log egress?
Define Desired PACS Network	100% private dedicated network or shared LAN? Is multiple facility connectivity required? Determine what new infrastructure is required
Consider Need to Modify Authentication Levels	Automatic change based on Duty Hours with manual option if Threat Level changes
Establish "Cut Over" Plan from Legacy to PIV System	Need a smooth transition from legacy cards to PIV credentials
Characterize Existing PACS Infrastructure	
Cabling	If CAT-5 or Equivalent, existing cabling can be utilized
Power Supplies	Can existing power supplies be utilized? (NOTE: PIV readers require more power than legacy technologies) Backup Power: are existing supplies sufficient to meet stated security goals for fail locked vs. fail unlocked?
Electrified Door Hardware	Quality door hardware should last 10 years under normal use and 5 years under heavy use.
Server and Client(s) Computers	A server should meet the requirements specified by the PACS vendors under consideration. Consider need for RAID and system back-up. Server should be dedicated to PACS and in installed in a secure room (may contain personal information.
Evaluate Existing PACS Software and Door Controllers	Is software modern, easy to use and reliable? How is existing service/support? Do existing PACS software and controllers have capability to be upgraded to match full FASC-N plus a minimum of 32 bits of public key certificate data? Does PACS software support a compatible "plug-in" that can perform CRL validation via OCSP or SCVP on a scheduled basis and automatically deny access to any credentials in the system that may be revoked?
User Enrollment	Define authentication mechanisms to be checked and/or enrolled: Name; FASC-N; PIV Expiration Date; Affiliation; PIV Certificate; CAK Certificate;

	Photograph; Biometric

BridgePoint PACS with PKI



HOW TO IMPLEMENT BRIDGEPOINT PACS WITH PKI	PROCEDURE/BENEFIT
1. AUTHENTICATE CREDENTIAL AND USER at ENROLLMENT Install TrustAlert Enrollment Station and Software that can Capture and Validate PIV Data Objects and Certificates from Credentials being Enrolled	Start by capturing certificates so that re-enrollment is not required at a later date. Strong Authentication at enrollment includes: - PIN Validation (the user is the owner) - BIOMETRIC Validation (the user is the owner) - Expiration Date Check (credential is not expired) - Display Photograph (Visual Check – Attended Access) - Verify Identical FASC-N Appears in all Data Objects - Validate Digital Signatures of Data Objects (Objects are signed by the Trusted Authority) - Private Key Challenge (PAK & CAK to verify private key public key pairs exist and card is not a clone)
2. IMPLEMENT PATH VALIDATION	Execute periodic revalidation of the full path to ensure all of the certificates in the path remain valid

At Enrollment, TrustAlert will Perform Path Validation to Insure Certificate(s) Have Not Been Revoked Continue with Periodic Path Validation Throughout the Enrollment Period	and have not been revoked. Set software to perform path validation every hour and to log results. Notify PACS System when a Certificate is Revoked.
3. INSTALL PKI SUPPORTED SOFTWARE AND CONTROLLER(S) Install New TrustZone PACS Software and Door Controller(s) that are PIV Compliant and Capable of Supporting PKI Operations Including an Interface to the TrustAlert Validation Service.	PACS Software and Controller(s) must read and store the entire FASC-N (or GUID) for interoperability and unique identification. Storing an additional 32 bits of data from the PKI Certificate can be used to establish integrity that a "rogue" certificate has not been introduced on a cloned PIV Credential that has already been enrolled.. There is a need to handle "errors in authentication." Keys Pairs that do not verify and Biometrics that do not match are reported by an 8-bit error message. TrustZone PACS will deactivate any Credential reported by TrustAlert to have a revoked certificate.
4. & 5. INSTALL TRUSTPOINT PKI AUTHENTICATING READER SOLUTION APPROPRIATE FOR DOOR BEING PROTECTED Dual Interface Readers are Capable of Operating in Contact and Contactless Modes. Challenge Card Auth Certificate (CAK private key) in 1F Contactless Mode. Challenge Personal Certificate (PAK private key) in 2F Contact Mode. Authentication Mechanisms Include 1-Factor (Card Only); 2-Factor (Card + PIN) and 3-Factor (Card + PIN + BIO).	Access Readers may be required in exterior locations where they could be subject to attack. In this case a Reader that only collects card data is preferred and the PKI Challenge – Response authentication is performed on the safe side of the door by the TrustPoint PKI Interface Module. In cases where there is not a fear of attack the PKI Authentication Readers can be mounted on the attack (exterior) side. Authentication Factors can be switched between 1F, 2F, and 3F based on time-of-day and threat level. Readers send FASC-N (GUID), any Error Message(s) and 32-bit Certificate Data to the PACS Controller for authorization check. Readers are equipped with Tamper Switch to warn of unauthorized removal from installed location.
6. PERFORM A FULL MATCH OF FASC-N, ERROR CODES AND PKI CERTIFICATE DATA	Match of 56 bit FASC-N assures Interoperability and Uniqueness. Match next 8 bits provides essential security information (bad BIO, bad private key verify). Match of 32 bits on PKI Certificate provides assurance that public key pairs with private key

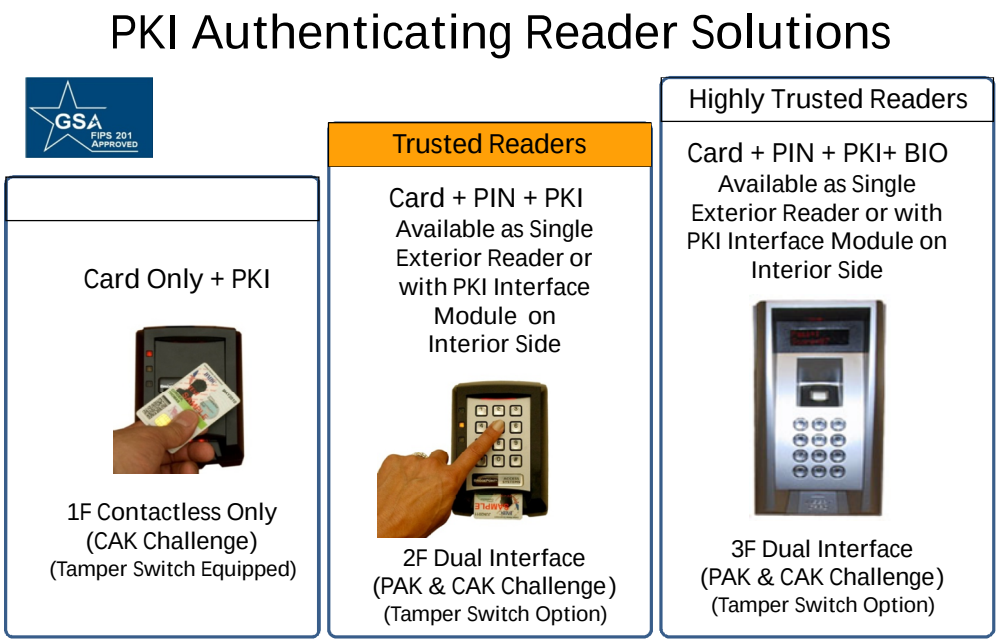
The PKI eFASC-N

It is possible to create a credential that bears all of the data from a genuine credential except for the private key. It is the challenge – response process that verifies the public – private key pair exist on the credential and establishes that the credential is not a clone.

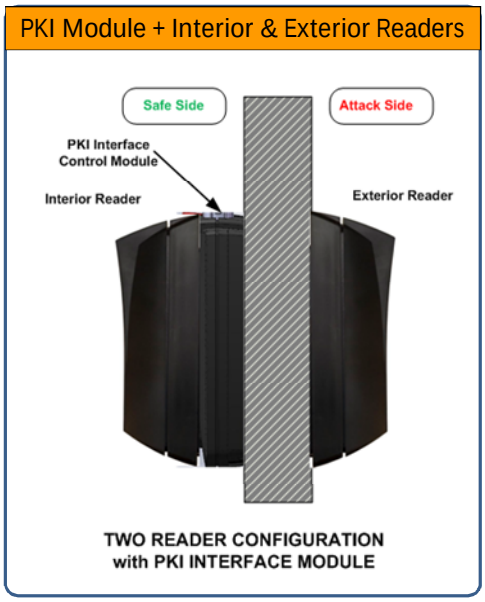
It is also possible to create a credential that bears the data from an authentic PIV and that also has a "rogue" public – private key pair that would verify even though the key pair would be different from the authentic credential. This means it would be possible to enroll an authentic credential into a PACS and then later use a copy of the credential that had a "rogue" key that would be granted access by the PACS if the certificate(s) had not appeared on the certificate revocation list (CRL).

This possibility can be mitigated by utilizing the public key certificate of the genuine credential in the matching process. The TrustZone PACS utilizes data from the certificate along with the FASC-N to create a “super” identifier called the “PKI enhanced FASC-N,” or “eFASC-N.” At enrollment, data from the genuine certificate (authenticated by PIN, BIO, Private Key Challenge and CRL Path Validation) is captured and stored in the system along with the FASC-N. When access is requested at the door, a PKI enabled reader performs a challenge – response to verify the key pair and then constructs and sends the eFASC-N to the system for matching of both data sets. With eFASC-N, a replica of a genuine credential would not be granted access because, while the key pair may verify, they would be different and therefore the certificate would be different from the certificate enrolled and the credential denied access.

The eFASC-N serves to bind PKI through the PACS from enrollment to access.



PKI Interface Control Module



BRIDGEPOINT™

Appendix 3 TrustBearer Visit Request Technical Specifications

Web-based digital certificate reading module for CAC and PIV/PIV-I

CertiPath requested the assistance of TrustBearer Labs to help develop the website where visit requests are submitted. TrustBearer provided its web browser smart card middleware, TrustBearer Live, corresponding service components, and a module for storing cardholder CAC & PIV certificates on the server.

This specification explains the requirements and TrustBearer products and services that were used to complete the task.

Project Requirements

Visit Request Web Application

CertiPath developed the visit request web application for the GSA Federated PACS demonstration that allows CAC & PIV cardholders to submit a visit request for a specific start and end date and time. This website has a calendar & time-picking user interface (https://visitrequest.exostar.com).

TrustBearer provided HTML, CSS and JavaScript code to detect and load its smart card middleware plug-in, TrustBearer Live, as well as sample code explaining how to read a user's certificate and store it to the server's file system with an associated visit start and end date & time stamp.

The web application runs on

- Microsoft Windows Server 2003
- Microsoft Internet Information Services (IIS) 6.0
- Microsoft ASP.NET 2.0

Supported Smart Cards

CAC, PIV and PIV-I smart cards are supported. This includes TWIC and FRAC credentials as they have a valid PIV application present on the credential.

Common Access Card (CAC) smart cards

The following table shows the current CAC platforms that are supported GSA Federated PACS demonstration by TrustBearer.

Attributes	**Current CAC Platforms**				
Card Manufacturer	Axalto	Gemplus	Axalto	Oberthur Card Systems (contactless pilot)	Oberthur Card Systems
Product Name	Axalto Cyberflex Access 64k v1 soft mask 4 version 1	GemXpresso (GXP) PRO 64 K	Axalto Cyberflex Access 64k v1 soft mask 4 version 2	ID-One Cosmo v5.2D 64K	ID-One Cosmo v5.2 72K

Laser Engraving on back of card	Axalto Access 64KV2	Gemplus GXP3 64V2N	Axalto Access 64KV2	Oberthur C.S. Cosmo64 V5.2D	Oberthur ID One V5.2
Chip Size (EEPROM)	64K	64K	64K	64K	72K
Answer to Reset (ATR)	3B751200002 905010401	3B6B00008065B 0830104748300 9000	3B751200002 905010401	3B DB 96 00 80 1F 03 00 31 C0 64 77E30300 82 9000 C1	3B DB 96 00 80 1F 03 00 31 C0 64 77E30300 82 9000 C1
CAC Applet Package	V2.3	V2.3 and V2.3.0c Applets	V2.3.0c Applets	V2.6.1 Applets	V2.6.1 Applets
Other features	Contact Only	Contact only	Contact only	Dual interface	Contact only
Operational Intro Date	April 2005	N/A	N/A	May 2006	January 2007
Operational Sunset Date	Q1 CY 2007	Q1 CY 2007	Q1 CY 2007	Q2 CY 2007	Ongoing
Expiration of last card issued	Q1 CY 2010	Q1 CY 2010	Q1 CY 2010	Q2 CY 2010	Ongoing

Attributes	Current CAC Platforms				
Card Manufacturer	Gemalto	Oberthur Card Systems (PIV Transitional)	Gemalto (PIV Transitional)	Oberthur Card Systems (PIV End-State) with PIV end-state applet and PIV auth cert	Gemalto (PIV End-State) with PIV end-state applet and PIV auth cert
Product Name	Cyberflex Access v2c 64K	ID-One Cosmo v5.2D 72K	Gemalto GemCombiXpresso R4 dual interface	ID-One Cosmo v5.2D 72K	Gemalto GemCombiXpresso R4 dual interface
Laser Engraving on back of card	Gemalto Access 64KV2	Oberthur ID One V5.2 Dual	Gemalto GCX4 72K DI	Oberthur ID One V5.2 Dual	Gemalto GCX4 72K DI
Chip Size (ROM/EEPROM)	64K	72K	72K	72K	72K
Answer to Reset (ATR)	3B959540FFAE01030000	3B DB 96 00 80 1F 03 00 31 C0 64 77E30300 82 9000 C1	3B 7D 96 00 00 80 31 80 65 B0 83 11 13 AC 83 00 90 00	3B DB 96 00 80 1F 03 00 31 C0 64 77E30300 82 9000 C1	3B 7D 96 00 00 80 31 80 65 B0 83 11 13 AC 83 00 90 00
CAC Applet Package	V2.6.1 Applets	V2.6.2 Applets	V2.6.2 Applets	V2.6.2 Applets	V2.6.2 Applets
Other features	Contact only	Dual interface	Dual interface	Dual interface	Dual interface
Operational Intro Date	February 2007	October 2006/ March 2007	June 2008	June 2008	June 2008
Operational Sunset Date	Ongoing	Ongoing	Ongoing	Ongoing	Ongoing
Expiration of last card issued	Ongoing	Ongoing	Ongoing	Ongoing	Ongoing

TrustBearer reads and stores the following free read information from the Common Access Card:

- Identity certificate
- Signature certificate
- Encryption certificate
- Card Authentication certificate (if present)
- CHUID (if present)

If the Identity certificate is not found, an error will be displayed to the user. If any of the other certificates are not found on the card, they will not be stored on the server.

Personal Identity Verification (PIV) and PIV-Interoperable (PIV-I) Smart Cards

TrustBearer supports all PIV cards that comply with the PIV Card Application AID and End-Point PIV Data Model as defined in NIST SP 800-73-2.

TrustBearer reads and stores the following free read information from PIV & PIV-I smart cards:

- PIV Authentication certificate
- Signature certificate (if present)
- Key Management certificate (if present)
- Card Authentication certificate (if present)
- CHUID

If the PIV Authentication certificate is not found, an error will be displayed to the user. If any of the other certificates are not found on the card, they will not be stored on the server.

TrustBearer Certificate Reading Module

TrustBearer provided a set of web browser & server components that enable digital certificates to be read from CAC, PIV and PIV-I smart cards and store those certificates to the server's file system with an identifier for the date & time of the requested visit.

Server Requirements

- Microsoft Windows Server 2003
- Microsoft Internet Information Services (IIS) 6.0 with ASP.NET support
- Microsoft .NET Framework 2.0
- Java Runtime Environment (JRE) version 6 with Java Cryptography Extension Unlimited Strength Jurisdiction Policy Files 6

Client Requirements

- Microsoft Windows XP or Vista (32-bit editions)
- Microsoft Internet Explorer 6 or 7
- Windows-compatible, PC/SC-compliant smart card reader (smart card reader driver to-be-provided by manufacturer)
- CAC, PIV or PIV-I smart card
- Administrative rights are not required to install the plug-in, but the user must be able to run digitally-signed executables downloaded from an Internet website.

Note: Mac OS X, Linux and Firefox plug-ins will be included, and should work, but are considered unsupported for this effort.

Custom Support for Federated PACS demonstration

The TrustBearer plug-in and server framework are available as commercial off the shelf components, but the certificate reader / writer module were custom developed for this project.

Required Features

- Certificate Reading: Reads X.509 digital certificates from the user's PIV, PIV-I or CAC smart card.

- Certificate Writing: After reading a CAC or PIV certificate, the certificate(s) will be written in DER format to a directory specified in a configuration file with a unique identifier, a type identifier and visit request time stamp as the filename.`[incrementing request id]_[certificate type]_[YYYYMMDDHHMM visit start time stamp]_[YYYYMMDDHHMM visit end time stamp].cer`CAC Certificate Types `CAC_ID : Identity (Required) CAC_SIG : Signature (optional) CAC_ENC`

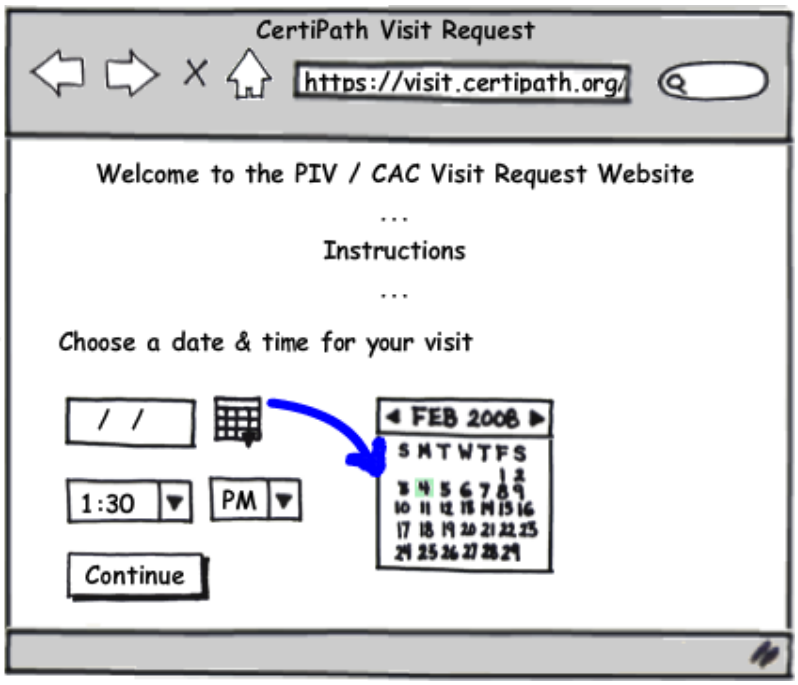
`: Encryption (optional)`PIV Certificate Types `PIV_AUTH : Authentication (Required) PIV_CARD_AUTH : Card Authentication (optional) PIV_SIG : Signature (optional) PIV_KEY_MGMT : Key Management (optional)`

- **CHUID Reading / Writing**: The binary data from the CHUID data container on PIV cards will be written to disk with a similar filename to the certificates. This data field is required. CHUIDs will be found on the PIV applet on PIV, PIV-I, CAC-NG and CAC-EP cards.`[incrementing request id]_PIV_CHUID_[YYYYMMDDHHMM visit start time stamp]_[YYYYMMDDHHMM visit end time stamp].bin`Example file names for the following 3 sequentially read cards containing all certificates and CHUIDs on PIV cards: PIV, then CAC, then PIV-I that are all requesting to visit on July 6, 2009 between 1:30 PM and 5:30 PM.
  ```
  21_PIV_AUTH_200907061330_200907061730.cer
  21_PIV_CARD_AUTH_200907061330_200907061730.cer
  21_PIV_SIG_200907061330_200907061730.cer
  21_PIV_KEY_MGMT_200907061330_200907061730.cer
  21_PIV_CHUID_200907061330_200907061730.bin
  22_CAC_ID_200907061330_200907061730.cer
  22_CAC_SIG_200907061330_200907061730.cer
  22_CAC_ENC_200907061330_200907061730.cer
  23_PIV_AUTH_200907061330_200907061730.cer
  23_PIV_CARD_AUTH_200907061330_200907061730.cer
  23_PIV_SIG_200907061330_200907061730.cer
  23_PIV_KEY_MGMT_200907061330_200907061730.cer
  23_PIV_CHUID_200907061330_200907061730.bin
  ```

- **Web Application**: A very simple, 2-page web-application will be provided that demonstrates steps 5-8 in the **Visit Request Workflow** use case. When accessing a TrustBearer web page, the query string variables ***visitStart*** and ***visitEnd*** must be defined with a 12-digit time stamp. Otherwise a message will be displayed to the user asking them to go back and pick a visit time before accessing this page.Query string variable format:...cert.html?**visitStart=YYYYMMDDHH00&visitEnd=YYYYMMDDHH00**where YYYY = 4 digit year (1900 - 9999) MM = 2 digit month (01 - 12) DD = 2 digit day (01 - 31) HH = 2 digit hour (00 - 23) MM = 2 digit minute (01 - 59)

CertiPath Visit Request Website Workflow

1. Cardholder accesses homepage of website
2. Cardholder reads instructions and picks a start and end date & time to visit the CertiPath / Exostar building.

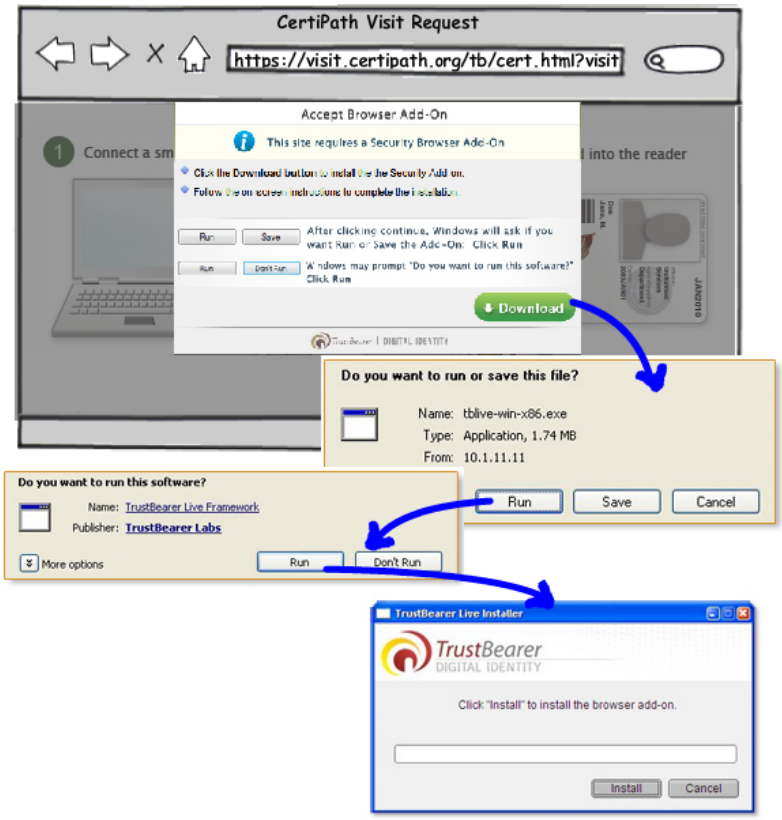


Note: CertiPath developed this portion of the website. Cardholder clicks “Continue” button.

3. Start and End date & time chosen are written to URL query string of TrustBearer certificate reader page, e.g. July 6, 2009 1:30 PM to 5:30 PM:https://visit.certipath.org/tb/cert.html?visitStart=200907061330&visitEnd=200907061730
4. This is where the TrustBearer certificate component takes over

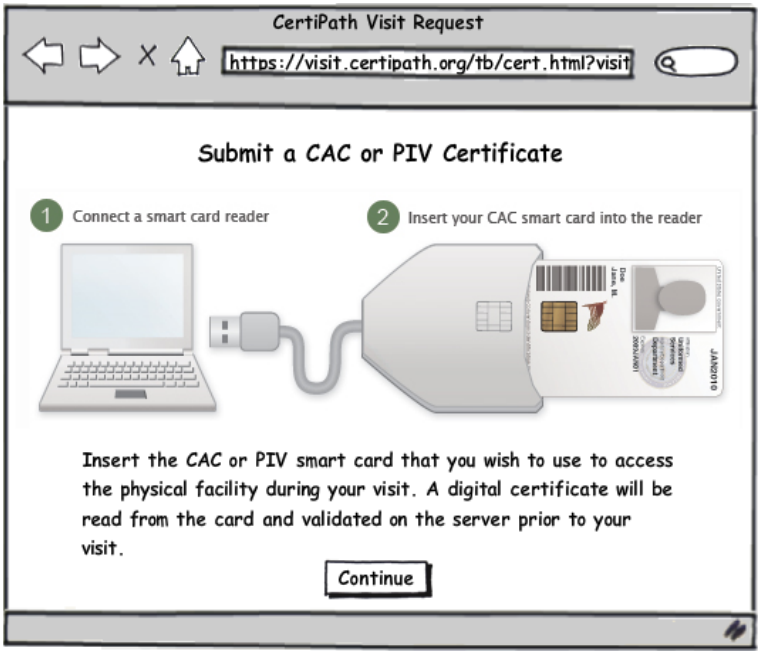
   If the user has not yet installed the TrustBearer browser add-on, a dialog will be displayed requesting the user to download and run the add-on. Once installation of the TrustBearer add-on is complete,

the workflow will continue.



If the user has previously installed the TrustBearer browser add-on, continue to the next step.

5. User will be prompted to connect a smart card reader, insert a CAC or PIV smart card, and click "Continue."



6. The appropriate certificate(s) & CHUID data will be read from the CAC or PIV smart card and written to the server's local storage drive. The certificate / CHUID file will be named: `[incrementing request id]_[certificate type]_[YYYYMMDDHHMM visit start time stamp]_[YYYYMMDDHHMM visit end time stamp].[cer|bin]` e.g. PIV card with all certificates except key management available,
   `9_PIV_AUTH_200907061330_200907061730.cer`
   `9_PIV_CARD_AUTH_200907061330_200907061730.cer`
   `9_PIV_SIG_200907061330_200907061730.cer`
   `9_PIV_CHUID_200907061330_200907061730.cer`

   If a smart card reader is not found, or if a compatible CAC or PIV card cannot be detected, the user will not be able to submit a request. TrustBearer's software will provide suggestions to resolving common problems.

Once the certificate & CHUID read is complete, a success message will be displayed to the end user.

Appendix 4 CoreStreet Communications Protocols

A detailed description of CoreStreet communications protocols used in the Federated PACS system is included below.

CoreStreet Reader Protocol

Specification

Version 1.4

30 Oct 2009

CoreStreet
One Alewife Center
Cambridge, MA 02140

Phone: 617 661 3554
Fax: 617 661 3542
www.corestreet.com

Table of Contents

Introduction ***3***

Communication Settings ***3***

Commands From Host to Reader/Card ***3***

Commands/Responses From Reader/Card to Host ***3***

Serial Protocol ***3***

Communication Flow ***4***

Serial Commands ***6***

HstCardPresent ***9***

HstPing ***11***

HstPINPresent ***12***

RdrBeginPoll ***13***

RdrEndPoll ***14***

RdrGetCapabilities ***15***

RdrGetCSRPVersion ***16***

RdrGetEnv ***17***

RdrHaltCard ***18***

RdrInfo ***19***

RdrIsCardPresent ***20***

RdrIsPINPresent ***21***

RdrMatchBio ***22***

RdrReboot ***23***

RdrResetRF ***24***

RdrSendExtendedCmd ***25***

RdrSetAV ***26***

RdrSetEnv ***28***

RdrTransceive ***29***

Copyright © 2009 CoreStreet, Ltd

RdrWaitForPIN ***30***

Copyright © 2009 CoreStreet, Ltd

1 Introduction

This document describes a generic serial interface to smart card / HID proximity card readers. By implementing the protocol described in this document, card reader vendors can seamlessly integrate their products into the CoreStreet product line.

2 Communication Settings

Full duplex, RS-485 / RS-422
Baud rate: Configurable to 9600, 19200, 38400 or 115200 via command card or extended command.

The reader must support:

- Cards that implement ISO14443-3 (e.g., Mifare).
- Cards that implement ISO14443-4 (T=CL), both Type A and Type B.
- HID proximity cards.

All card communications must be accessible via the reader's serial port interface.

3 Commands From Host to Reader/Card

Host-to-reader commands fall into two classes: card commands or reader commands. Card commands are sent directly to the card and data is returned directly from the card (wrapped with the appropriate protocol bytes). Reader device commands are interpreted by the reader itself. The reader may act on these commands on its own or it may issue commands to the connected card to generate a response.

4 Commands/Responses From Reader/Card to Host

Communications from the reader to the host can be initiated by the reader (reader-to-host commands) or sent as a response to a host-initiated command. Reader responses always follow an issued command. Reader-initiated communication occurs on its own.

5 Serial Protocol

Every command (other than the HstPing command) requires a response. If no data is returned for a given command, then a simple status response is generated.

Each message must be layered with control data to aid in the delivery of complete and accurate messages. The wrapped commands are defined as:

```
DLE STX CMD LEN-MSB LEN-LSB DATA[0] . . . DATA[LEN-1] CHK DLE ETX
```


Copyright © 2009 CoreStreet, Ltd

The wrapped responses are defined as:

`DLE STX RSP LEN-MSB LEN-LSB STA DATA[0] . . . DATA[LEN-1] CHK DLE ETX`

(each item is 1 byte in length)

Where:

> `CMD` is the command (see Table 1 and Table 2 below)
>
> `RSP` is the response "command" (i.e., 0x40)
>
> `STA` is the status of the response (see Table 3 below)
>
> `LEN` is the 2 byte value made up of `LEN-MSB` and `LEN-LSB` and is defined as the `DATA` length only.
>
> `CHK` is defined as the XORing of each byte between `STX` and `CHK` (i.e `CMD`, `LEN-MSB, LEN-LSB`, `DATA[0] … DATA[LEN-1]`).
>
> `STX` = 0x02
>
> `ETX` = 0x03
>
> `DLE` = 0x10

If any character in the `DATA` stream or checksum is equal to the `DLE` value, it should be sent twice (`DLE` stuffed), but the additional DLE must not be counted in the length or in the checksum.

5.1.1 Communication Flow

Either the host or the reader may initiate communication. However, once the reader sends the HstCardPresent command to the host, it must only send responses to host commands until it receives the RdrHaltCard command, i.e. it must not initiate any communication. A typical communication session will look like this:

Host	**Reader**	**Notes**
RdrBeginPoll		Reader begins polling for cards
	Response STA_OK	
	HstPing	
	HstPing	
	HstPing	
	HstCardPresent	Card presented to reader (polling now suspended)
Response STA_OK		
RdrTransceive		

Copyright © 2009 CoreStreet, Ltd

	Response STA_OK	
RdrIsCardPresent		Is the card still there?
	Response STA_OK	
RdrTransceive		
	Response STA_OK	
RdrHaltCard		Host is done with the card – reader may initiate commands again (reader resumes polling for cards)
	Response STA_OK	
	HstPing	
	HstPing	
	Etc…	
RdrEndPoll		Reader stops polling for cards
	Response STA_OK	

Copyright © 2009 CoreStreet, Ltd

6 Serial Commands

Table 1: Command Set (Host to Reader)

Command Name	CMD Code	Len	Data	Resp Data Len	Resp Data
RdrBeginPoll	0x20	var	List of card types	0	N/A
RdrEndPoll	0x21	0	N/A	0	N/A
RdrMatchBio	0x22	var	Accuracy threshold (1 byte) + bio templates (var)	1	0x00 for no match, 0x01 for template1 match, 0x02 for template2 match
RdrGetEnv	0x23	var	Tag	var	Env data
RdrHaltCard	0x24	0	N/A	0	N/A
RdrInfo	0x25	0	N/A	var	Reader name, Firmware version, and any other reader details the manufacturer wishes to provide
RdrIsCardPresent	0x26	0	N/A	2	0x00 for no card or card type for card in field from Table 4*, followed by interface type (0x00 for unknown)
RdrReboot	0x27	2	Delay in ms	0	N/A
RdrResetRF	0x28	0	N/A	0	N/A
RdrSetAV	0x29	var	LED/buzzer/LCD bit flags + audio duration in ms (2 bytes) + 1 byte RFU + ASCII text (optional)	0	N/A
RdrSetEnv	0x2A	var	Tag=Value	0	N/A
RdrGetCSRPVersion	0x2B	0	N/A	2	Two-byte major/minor version of the supported CSRP
RdrGetCapabilities	0x2C	0	N/A	2	Two-byte capabilities bitmask
RdrSendExtendedCmd	0x2D	var	var	var	var
RdrIsPINPresent	0x2E	0	N/A	var	PIN data
RdrTransceive	0x30	var	T=CL /Mifare command + params	var	T=CL/Mifare response data
RdrWaitForPIN	0x31	0	N/A	var	PIN data with STA_OK or STA_TIMEOUT

Copyright © 2009 CoreStreet, Ltd

Table 2: Command Set (Reader to Host)

Command Name	CMD Code	Len	Data	Resp Data Len	Resp Data
HstPing	0xA0	0	N/A	0	N/A
HstCardPresent	0xA1	var	Card type from Table 4*, plus n number of bytes from the card as described in the command details section of this doc	0	N/A
HstPINPresent	0xA2	var	PIN digits (excluding terminating character - #, Enter, etc.)	0	N/A

Table 3: Status Codes

Status Name	STA Code	Description
STA_OK	0x00	Success
STA_NO_CARD	0x01	Card no longer in the field
STA_BAD_CMD	0x02	Unknown command
STA_BAD_FORMAT	0x03	Bad command format
STA_CHK_ERR	0x04	Checksum error
STA_GEN_ERR	0x05	General error
STA_CARD_COMM_ERR	0x06	Card communication error
STA_TIMEOUT	0x07	Timeout while entering PIN

Table 4: Card Types

Card Type Name	Card Type Code
Mifare Classic	0x01
DESFire	0x02
PIV	0x03
HID Prox	0x04
ISO 14443 A Generic	0x05
ISO 14443 B Generic	0x06
ISO 7816	0x07
TWIC	0x08
CAC / CAC-NG	0x09

Copyright © 2009 CoreStreet, Ltd

Table 5: Environment Variables

Variable Name	Description
KPADBYTES	Maximum number of keypad bytes that will be returned by the reader to the host. Default is 16 bytes.
KPADMODE	0 = PIN + Card **OR** Card + PIN 1 = PIN + Card 2 = Card + PIN Default is 0.
KPADTIMEOUT	Maximum number of seconds the user has to enter data on the keypad before the buffer is reset. A value of 0 instructs the reader that the keypad is not being used. Default is 5 seconds. NOTE: This is the timeout between key presses, not the overall time allotted for PIN entry.
NBYTES	Number of bytes to return for Mifare or Desfire. Default is 16 bytes.
MIFSTART	Location block of the first byte of data to return for Mifare cards. Default is 1.

* NOTE About Card Detection: If both an HID Prox card and a smart card are in the reader field at the same time, the reader should ignore the HID Prox card. This is important for legacy implementations where people must carry legacy badges as well as their smart cards.

Table 6: Reader Interface Types

Reader Interface Type Name	Reader Interface Type Code
Unknown	0x00
Contact	0x01
Contactless	0x02

Copyright © 2009 CoreStreet, Ltd

6.1.1 HstCardPresent

The HstCardPresent command informs the host that a card is in the reader's RF field. The returned data varies based on the card type.

Table 7: Variable Return Data From the HstCardPresent Command

Card Name	Card Type	Fields Returned	Field Data Description
Mifare Classic	0x01	T I U S L A D	T = 1 byte Card Type I = 1 byte Reader Interface Type U = 4 byte UID S = 1 byte SAK L = len of ATS A = L byte(s) ATS D = n bytes of data starting at block b
DESFire	0x02	T I U S L A D	T = 1 byte Card Type I = 1 byte Reader Interface Type U = 7 byte UID S = 1 byte SAK L = len of ATS A = L byte(s) ATS D = first n bytes of application data
PIV Application	0x03	T I U	T = 1 byte Card Type I = 1 byte Reader Interface Type U = 25 byte FASC-N
HID Prox	0x04	T I U	T = 1 byte Card Type I = 1 byte Reader Interface Type U = 6 bytes of Wiegand Data right justified - Preceded by a sentinel bit. 1st transmitted wiegand bit is the 1st bit past the sentinel bit.
Generic 14443A	0x05	T I	T = 1 byte Card Type I = 1 byte Reader Interface Type
Generic 14443B	0x06	T I	T = 1 byte Card Type I = 1 byte Reader Interface Type
ISO 7816	0x07	T I	T = 1 byte Card Type I = 1 byte Reader Interface Type
TWIC	0x08	T I U	T = 1 byte Card Type I = 1 bytes Reader Interface Type U = 25 byte FASC-N
CAC / CAC-NG	0x09	T I U	T = 1 byte Card Type I = 1 byte Reader Interface Type U = 40 byte SEIWG data

- See Table 4 for card types
- See Table 5 for environment variables to set "n" and "b"
- See Table 6 for reader interface types

Copyright © 2009 CoreStreet, Ltd

Example 1 – Mifare Classic card:

Command From Reader to Host:

```
10 02 A1 00 11 01 99 88 77 66 DD 02 88 88 11 22
33 44 55 66 77 88 E6 10 03
```

Response From Host to Reader:

```
10 02 40 00 00 00 40 10 03
```

The reader has sent a HstCardPresent command: `A1`
There are `00 11` bytes of data.
The first byte of data shows a Mifare classic card: `01`.
The UID is `99 88 77 66`.
The SAK is `DD`.
The ATS is `02` bytes long
The ATS is `88 88`.
The reader was configured to send 8 bytes of app data: `11 22 33 44 55 66 77 88`.
The checksum is `E6`.
(XOR `A1 00 11 01 99 88 77 66 DD 02 88 88 11 22 33 44 55 66 77 88`).

Example 2 - Mifare Classic Card with DLE Stuffing:

Command From Reader to Host:

```
10 02 A1 00 11 01 99 88 77 10 10 DD 02 88 10 10
11 22 33 44 55 10 10 77 88 7E 10 03
```

Response From Host to Reader:

```
10 02 40 00 00 00 40 10 03
```

The reader has sent an HstCardPresent command: `A1`
There are `00 11` bytes of data.
The first byte of data shows a Mifare classic card: `01`.
The UID is `99 88 77 10`.
The SAK is `DD`.
The ATS is `02` bytes long
The ATS is `88 10`.
The reader was configured to send 8 bytes of app data: `11 22 33 44 55 10 77 88`.
The checksum is `7E`.
(XOR `A1 00 11 01 99 88 77 10 DD 02 88 10 11 22 33 44 55 10 77 88`).

Note the DLE stuffing that the reader has inserted due to data bytes that happened to equal DLE. Note also that these “stuffed” bytes were NOT included in the length or in the checksum.

Copyright © 2009 CoreStreet, Ltd

6.1.2 HstPing

The HstPing command informs the host that the reader is awake and functioning properly. This command must be sent within 5 seconds of the last reader to host communication. Any command or response sent by the reader should reset the timer for the HstPing command. Unlike all other commands, this command has no response.

Example:

Command From Reader to Host:

```
10 02 A0 00 00 A0 10 03
```

Response From Host to Reader:

```
None
```


Copyright © 2009 CoreStreet, Ltd

6.1.3 HstPINPresent

The HstPINPresent command informs the host that the cardholder entered a PIN via the reader's keypad. The data that is returned varies based on the card type.

Keypad data is delimited by pressing the # key on the keypad (Card + PIN + "#") or by presenting a card after entering data (PIN + Card or PIN + "#" + Card). The maximum number of keypad bytes returned by the reader is set in the KPADBYTES environment variable (the default is 16 bytes).

The reader's response to this command contains the keypad data in ASCII format, excluding the PIN termination character (typically the '#' key), if any.

Example: Reader is instructed to return keypad data and reader responds

Command From Reader to Host:

```
10 02 A2 00 08 31 32 33 34 35 36 37 38 A2 10 03
```

Response From Host to Reader:

```
10 02 40 00 00 00 40 10 03
```

The reader has sent a HstPINPresent command: `A2`
There are `00 08` bytes of data.
The keypad data: `31 32 33 34 35 36 37 38`.
The checksum is `A2.`
(XOR `A2 00 08 31 32 33 34 35 36 37 38).`

Copyright © 2009 CoreStreet, Ltd

6.1.4 RdrBeginPoll

The RdrBeginPoll command instructs the reader to begin polling for cards placed in the reader's field. The reader will report a card in the field if it matches one of the types specified in this command. The card types are each single byte values and should be specified one after another, without a delimiter. Order does not matter.

Example: Instruct the reader to poll for Mifare Classic, DESFire and HID Prox cards

Command From Host to Reader:

```
10 02 20 00 03 01 02 04 24 10 03
```

Response From Reader to Host:

```
10 02 40 00 00 00 40 10 03
```

The host has sent a RdrBeginPoll command: `20`
There are `00 03` bytes of data.
The reader has been instructed to poll for Mifare Classic (01), DESFire (02) and HID Prox cards (04).
The checksum is `24` (XOR `20 00 03 01 02 04`).

This command returns no data from the reader, so the reader simply responds with a STA_OK result and begins polling.

Copyright © 2009 CoreStreet, Ltd

6.1.5 RdrEndPoll

The RdrEndPoll command instructs the reader to stop polling for cards. After receiving this command the reader will no longer report that any cards are present in its field. The only communication from reader to host will be via the HstPing command.

Example: Instruct the reader to stop polling for cards

Command From Host to Reader:

```
10 02 21 00 00 21 10 03
```

Response From Reader to Host:

```
10 02 40 00 00 00 40 10 03
```

The host has sent a RdrEndPoll command: `21`
There are `00 00` bytes of data.
The checksum is 21 (XOR `21 00 00`).

This command returns no data from the reader, so the reader simply responds with a STA_OK result and stops polling.

Copyright © 2009 CoreStreet, Ltd

6.1.6 RdrGetCapabilities

The RdrGetCapabilities command instructs the reader to send a two-byte bitmask enumerating its capabilities (i.e., keypad, LCD, etc.)

Capabilities (OR'ed together by reader to form two-byte bitmask):

Red LED	0x0001
Green LED	0x0002
Yellow/Orange LED	0x0004
Buzzer 1	0x0008
Buzzer 2	0x0010
Buzzer 3	0x0020
LCD	0x0040
Keypad	0x0080
Bio sensor	0x0100
Contactless Reader	0x0200
Contact Reader	0x0400

Example: Query the reader for its capabilities

Command From Host to Reader:

`10 02 2C 00 00 2C 10 03`

Response From Reader to Host:

`10 02 40 00 02 00 XX XX ZZ 10 03`

The host has sent a RdrGetCapabilities command: `2C`
There are `00 00` bytes of data.
The checksum is `2C` (XOR `2C 00 00`).

This command returns the reader's capabilities:
Data Length: `00 02`
STA_OK: `00`
Two byte capabilities bitmask (MSB LSB): `01 4B`
The checksum is 42 (XOR `40 00 02 00 01 01`).

This response indicates that the reader has a red LED, green LED, buzzer 1, LCD, keypad, and bio sensor.

Copyright © 2009 CoreStreet, Ltd

6.1.7 RdrGetCSRPVersion

The RdrGetCSRPVersion command instructs the reader to send the major and minor version of the CSRP that it implements (as specified on the first page of this document).

Example: Query the reader for the version of the CSRP it implements

Command From Host to Reader:

```
10 02 2B 00 00 2B 10 03
```

Response From Reader to Host:

```
10 02 40 00 02 00 01 01 42 10 03
```

The host has sent a RdrGetCSRPVersion command: `2B`
There are `00 00` bytes of data.
The checksum is 2B (XOR `2B 00 00`).

This command returns the version of the CSRP that the reader implements:
STA_OK: `00`
One byte CSRP major version: `01`
One byte CSRP minor version: `01`
The checksum is 42 (XOR `40 00 02 00 01 01`).

Copyright © 2009 CoreStreet, Ltd

6.1.8 RdrGetEnv

The RdrGetEnv command instructs the reader to return the value for the given variable. If there is no variable matching the name provided, the reader must return a zero length response with STA_OK. NOTE: Environment variables are case-sensitive.

Example: Query the reader for the value of the NBYTES variable

Command From Host to Reader:

```
10 02 23 00 06 4E 42 59 54 45 53 32 10 03
```

Response From Reader to Host:

```
10 02 40 00 02 00 31 36 45 10 03
```

The host has sent a RdrGetEnv command: `23`
There are `00 06` bytes of data.
The data sent is `4E 42 59 54 45 53` (hex value for NBYTES)
The checksum is 32 (XOR `23 00 06 4E 42 59 54 45 53`).

This command returns the keypad data from the reader:
STA_OK: `00`
Two bytes of enviornment data: `02`
The environment data: `31 36` (NOTE: this is NOT numeric data, it's text, in this case "16")
The checksum is `45` (XOR 40 `00 02 00 31 36`).

Note: Environment variables always store text data

Copyright © 2009 CoreStreet, Ltd

6.1.9 RdrHaltCard

The RdrHaltCard command instructs the reader to stop communicating with the current card in the field. The reader should not send any HstCardPresent commands until the current card in the field has been removed and put back in the field, or a new card enters the field.

Until the reader sees this command for the last reported card in the field, it should not send serial data unless it is a response to a host based command. The host will always send the RdrHaltCard message when it has finished processing a card – even if it encounters an error while processing.

Example:

Command From Host to Reader:

`10 02 24 00 00 24 10 03`

Response From Reader to Host:

`10 02 40 00 00 00 40 10 03`

Explanation:

The host has sent the following:

RdrHaltCard Command: `24`
Zero additional command bytes: `00 00`
Checksum: 24

It has received a successful response from the reader.

Copyright © 2009 CoreStreet, Ltd

6.1.10 RdrInfo

The RdrInfo command instructs the reader to send version information back to the host.

Example: Host queries reader for its version information

Command From Host to Reader:

```
10 02 25 00 00 25 10 03
```

Response From Reader to Host:

```
10 02 40 XX XX 00 YY YY … YY YY ZZ 10 03
```

Explanation:

The host has sent the following:

RdrInfo Command: `25`
Zero additional command bytes: `00 00`
Checksum: `25`

It has received a successful response from the reader (STA_OK). The reader may reply with any version information that it wants to, as long as the message conforms to the serial specification. It must return the number of bytes and a valid checksum for the message.

Data Length: `XX XX`
STA_OK: `00`
Version information: `YY YY … YY YY`
Checksum: `ZZ`

Copyright © 2009 CoreStreet, Ltd

6.1.11 RdrIsCardPresent

The RdrIsCardPresent command asks the reader if there is a card in the reader's field. If no card is present, then the response data will be 00 00. Otherwise, the response data will have the one byte card type value for the card in the field and the one byte interface type.

Example: Host asks reader if a card is present. Reader has DESFire card in its field.

Command From Host to Reader:

`10 02 26 00 00 26 10 03`

Response From Reader to Host:

`10 02 40 00 02 00 02 01 41 10 03`

Explanation:

The host has sent the following:

RdrIsCardPresent Command: `26`
Zero additional command bytes: `00 00`
Checksum: `26`

It has received a successful response from the reader.

Data Length: `00 02`
STA_OK: `00`
Card Type (DESFire): `02`
Interface Type (Contact): `01`
Checksum: `41`

Copyright © 2009 CoreStreet, Ltd

6.1.12 RdrIsPINPresent

The RdrIsPINPresent command asks the reader if the user has entered a PIN at the reader. If no PIN is present (either because the user did not enter one or because the KEYPADTIMEOUT elapsed), the reader must return a zero length response with STA_OK. Otherwise, the response data consists of the PIN in ASCII format.

Example: Host asks reader if a PIN is present. Reader has a PIN in its buffer.

Command From Host to Reader:

`10 02 2E 00 00 2E 10 03`

Response From Reader to Host:

`10 02 40 00 08 31 32 33 34 35 36 37 38 40 10 03`

Explanation:

The host has sent the following:

RdrIsPINPresent Command: `2E`
Zero additional command bytes: `00 00`
Checksum: `2E`

It has received a successful response from the reader.

Data Length: `00 08`
STA_OK: `00`
PIN data: `30 31 32 33 34 35 36 37 38`
Checksum: `40`

Copyright © 2009 CoreStreet, Ltd

6.1.13 RdrMatchBio

The RdrMatchBio command instructs the reader to perform a biometric match using the cardholder's presented biometric factor and the biometric templates stored on the card. The accuracy threshold is a single byte value that indicates how accurate the match must be in order to return a successful match. The biometric templates (in INCITS-378 format) follow the accuracy threshold byte.

Accuracy threshold values:

Low	0x01
Medium	0x02
High	0x03

The response from the host indicates whether or not the match was successful. A response of 0x00 indicates no match, a response of 0x01 indicates a match with the first template, and a response of 0x02 indicates a match with the second template.

Example: Instruct the reader to perform a biometric match with high accuracy

Command From Host to Reader:

```
10 02 22 XX XX 03 YY YY . . . YY ZZ 10 03
```

Response From Reader to Host:

```
10 02 40 00 01 00 02 43 10 03
```

The host has sent a RdrMatchBio command: `22`
There are `XX XX` bytes of data.
The reader has been instructed to use high accuracy (`03`) for the match.
Biometric templates: YY YY . . . YY
The checksum is ZZ.

The response from the reader indicates that the presented biometric factor matched the second template stored on the card.

Copyright © 2009 CoreStreet, Ltd

6.1.14 RdrReboot

The RdrReboot command instructs the reader to reboot. The reader responds to the command and reboots.

Example:

Command From Host to Reader:

`10 02 27 00 00 27 10 03`

Response From Reader to Host:

`10 02 40 00 00 00 40 10 03`

Explanation:

The host has sent the following:

RdrReboot Command: `27`
Zero additional command bytes: `00 00`
Checksum: `27`

It has received a successful response from the reader. The reader now reboots.

Copyright © 2009 CoreStreet, Ltd

6.1.15 RdrResetRF

The RdrResetRF command instructs the reader to reset its RF field and report on any cards in the field. If the reader had "halted" a card in the field, it will no longer be "halted." This command should not completely reboot the reader. The reader should only reset the reader's RF field. The reader should generate a responds after resetting the RF field.

Example:

Command From Host to Reader:

`10 02 28 00 00 28 10 03`

Response From Reader to Host:

`10 02 40 00 00 00 40 10 03`

Explanation:

The host has sent the following:

RdrResetRF Command: `28`
Zero additional command bytes: `00 00`
Checksum: `28`

The reader has reset the RF field and has sent a successful response to the host.

Copyright © 2009 CoreStreet, Ltd

6.1.16 RdrSendExtendedCmd

The RdrSendExtendedCmd command instructs the reader to perform a reader-specific command (i.e., a command outside the scope of the CSRP). Upon receiving the command, the reader must carry it out and then respond with its reader-specific response. The command and response are formatted like all standard CSRP commands/responses, but the data payload is reader-specific.

Example: Instruct the reader to carry out an extended command

Command From Host to Reader:

```
10 02 2D XX XX YY YY . . . YY ZZ 10 03
```

Response From Reader to Host:

```
10 02 40 XX XX STA YY YY . . . YY ZZ 10 03
```

The host has sent a RdrSendExtendedCmd command: `2D`
There are `XX XX` bytes of data.
The checksum is ZZ.

Copyright © 2009 CoreStreet, Ltd

6.1.17 RdrSetAV

The RdrSetAV command sends Audio/Visual commands to the reader. The host uses this command to:

- Change the LED status of the reader
- Sound the reader's buzzer(s)
- Modify its LCD display (if present).

By providing a bit flag and audio duration, the host can control multiple settings in one call.

NOTE: It is important for the reader to process the RdrSetAV command in the background; RdrSetAV must never delay the processing of additional commands.

The reader's initial state on startup is:

Red LED on
All other LEDs off
Buzzer off
LCD off

Bit values:

All off	0x00
Red LED	0x01
Green LED	0x02
Yellow/Orange LED	0x04
Buzzer 1	0x08
Buzzer 2	0x10
Buzzer 3	0x20
LCD	0x40

NOTE: Not all options are available on all readers. Multiple buzzer flags are defined if the reader allows controlling the pitch of the buzzer.

The audio duration is specifed in milliseconds and is supplied as a 2 byte value. The audio duration only applies to the buzzers; if the caller doesn't specify any buzzer bits, the reader ignores the audio duration setting.

The fourth byte is reserved for future use (RFU) and must be set to zero.

If the LCD bit is set, the reader treats all data following the RFU byte as ASCII text to display on the LCD. If the LCD bit is not set, the reader ignores any data following the RFU byte.

Example 1: Turn on the green LED and sound buzzer 1 for 200 ms (turns off all other A/V components)

Command From Host to Reader:

```
10 02 29 00 04 0A 00 C8 00 EF 10 03
```


Copyright © 2009 CoreStreet, Ltd

Response From Reader to Host:

```
10 02 40 00 00 00 40 10 03
```

Explanation:

<u>The host has sent the following:</u>
RdrSetAV Command: `29`
Four command bytes: `00 04`
Bit flags – Green + Buzzer 1: `0A`
Duration – 200 ms: `00 C8`
RFU byte: `00`
Checksum: `EF (XOR 29 00 04 0A 00 C8 00)`

It has received a successful response from the reader.

Example 2: Display "Access Granted" on the LCD (turns off all other A/V components)

Command From Host to Reader:

```
10 02 29 00 12 40 00 00 00 41 63 63 65 73 73 20
47 72 61 6E 74 65 64 30 10 03
```

Response From Reader to Host:

```
10 02 40 00 00 00 40 10 03
```

Explanation:

<u>The host has sent the following:</u>
RdrSetAV Command: `29`
Eighteen command bytes: `00 12`
Bit flags – LCD: `40`
Duration – 2000 ms: `07 D0`
RFU byte: `00`
ASCII Text (Access Granted): `41 63 63 65 73 73 20 47 72 61 6E 74 65 64`
Checksum: `30 (XOR 29 00 12 40 00 00 00 41 63 63 65 73 73 20 47 72 61 6E 74 65 64)`

It has received a successful response from the reader.

Copyright © 2009 CoreStreet, Ltd

6.1.18 RdrSetEnv

The RdrSetEnv command instructs the reader to set an environment variable. The currently supported environment variables are shown in Table 5.

Example: Set NBYTES to 32

Command From Host to Reader:

```
10 02 2A 00 09 4E 42 59 54 45 53 3D 33 32 08 10 03
```

Response From Reader to Host:

```
10 02 40 00 00 00 40 10 03
```

Explanation:

The host has sent the following:

RdrSetEnv Command: `2A`
Nine additional command bytes: `00 09`
Command Data (NBYTES=32): `4E 42 59 54 45 53 3D 33 32`
Checksum: `08 (XOR 2A 00 09 4E 42 59 54 45 53 3D 33 32)`

The host has received a successful response from the reader.

Copyright © 2009 CoreStreet, Ltd

6.1.19 RdrTransceive

The RdrTransceive command transmits raw commands to the card in the field and receives the raw data that is returned by the card. The transmitted commands may be either Mifare Classic commands or T=CL commands.

Example - Send a Load Key Command to a Mifare Classic Card:

Command From Host to Reader:

```
10 02 30 00 0F 4C 00 00 00 00 00 00 00 00 DE AD
BE EF BA BE 6A 10 03
```

Response From Reader to Host:

```
10 02 40 00 00 00 40 10 03
```

Explanation:

The host sends the reader:

RdrTransceive: `30`
15 bytes command: `00 0F`
Mifare Load Key: `4C`
Mode: `00`
KeyNo: `00`
Transfer Key: `00 00 00 00 00 00`
Read/Write Key: `DE AD BE EF BA BE`
Checksum: `6A`

This command returns no data from the card, so the reader simply responds with a STA_OK result.

Copyright © 2009 CoreStreet, Ltd

6.1.20 RdrWaitForPIN

The RdrWaitForPIN command is a blocking call which instructs the reader to get the PIN from the keypad, and return it to the host. The reader must use the environment variables: KPADBYTES, KPADTIMEOUT, and KPADMODE to determine how the PIN should be entered.

The reader must return the PIN in ASCII with a status of STA_OK once KPADBYTES bytes of data has been entered, or the PIN delimeter (#) has been pressed before KPADTIMEOUT has elapsed. NOTE: KPADTIMEOUT is the maximum time the reader allows between key presses, not the overall PIN entry time.

If KPADTIMEOUT seconds elapses between key presses, the reader must return a zero length PIN with the status code STA_TIMEOUT.

Example – User Enters 8 Digit PIN + "#":

Command From Host to Reader:

`10 02 31 00 00 31 10 03`

Response From Reader to Host:

`10 02 40 00 08 31 32 33 34 35 36 37 38 40 10 03`

Explanation:

The host has sent the following:

RdrWaitForPIN Command: `31`
Zero additional command bytes: `00 00`
Checksum: `31`

It has received a successful response from the reader.
Data Length: `00 08`
STA_OK: `00`
PIN data: `30 31 32 33 34 35 36 37 38`
Checksum: `40`

Copyright © 2009 CoreStreet, Ltd