

**Deploying SQL Server 2008 Based on Payment
Card Industry Data Security Standards
(PCI DSS)**

Version 1.2

A White Paper by

John Bastow, Caturano and Company PC

Cindy Papas, ParenteBeard LLC

Table of Contents

Executive Summary	3
1. Overview	4
1.1 Payment Card Industry Data Security Standards Background	4
1.2 SQL Server Today	4
1.3 PCI Requirements and SQL Server 2008	5
2. SQL Server 2008 Features in Use	7
2.1 Transparent Data Encryption	7
2.1.1 Extensible Key Management	8
2.1.2 PCI DSS Key Management Requirements	9
2.1.3 Other Encryption Considerations	10
2.2 SQL Server Audit	10
2.3 Enhanced SQL Server 2008 Access Security Features	14
2.3.1 Signed Module	14
2.3.2 Windows Authentication	15
2.3.3 BUILTIN/Administrators Group	16
2.3.4 Role Based Access and Other Access Control Considerations	17
2.4 Default SQL Server 2008 Features	17
2.5 Policy-Based Management	19
2.5.1 Encryption Policy	20
2.5.2 SQL Server Audit Policy	21
2.5.3 Role Based Access Policy	22
2.5.4 Authentication Policy	22
2.5.5 Factory Default Settings Policy	23
3. Conclusion	24
3.1 Resources	24

Executive Summary

With technology changes occurring at a rapid pace, IT professionals are continually challenged with the need to balance scalability, compatibility and ease of maintenance to control areas such as security administration and transaction monitoring. Recent data breaches have forced database managers and Chief Information Officers to consider controls to mitigate risks as technology is installed and not as an afterthought. Guidelines from government and industry require a new approach to implementing systems and databases. Organizations that consider compliance as part of their implementation and design stand a better chance of mitigating reputation, security and fraud risk before it can occur.

One of the standards that developers of Microsoft SQL Server 2008 should consider as part of their system implementation is the Payment Card Industry Data Security Standards (PCI DSS). Since 2005, more than 252 million credit card records have been breached. The increase in data security breaches has prompted and mandated the development of the PCI DSS and other regulations to protect the security of cardholder data. The standards were developed to address the need for strengthening security over sensitive cardholder data elements and providing a foundation for bolstering technical and operational security control activities through a combination of preventive and detective controls and continuous vigilance. The standards consist of twelve data security requirements supporting six control objectives. If you are a company that stores, processes and/or transmits sensitive cardholder data, you are required to demonstrate compliance with PCI DSS.

The key to complying with the standards is to ensure that Information Technology professionals maintain a suitable database platform to allow requirements to be met. Technology professionals need to ensure that they are implemented in an automated and controlled fashion to ensure effective transaction processing. Due to the popularity and proven reputation of SQL Server, critical applications continue to be installed on such highly dependable database software. SQL Server 2008 offers robust new features that provide the means for meeting multiple PCI DSS requirements to properly support key production systems within the organization's environment.

The purpose of this white paper is to provide developers and senior technology leaders with technical solutions on how to proactively achieve PCI compliance when deploying SQL Server 2008 to support and protect key business processes within an organization and avoid the risks noted above.

1. Overview

1.1 Payment Card Industry Data Security Standards Background

Today, billions of people use payment cards to settle sales transactions. In 2004, five key payment card brands – American Express, Discover Financial Services, JCB International, MasterCard Worldwide and Visa International – joined together as founding members to establish a common set of information security requirements. The PCI DSS were developed to protect cardholder information and mitigate the risk of loss or theft of payment card related information. The objective of the PCI DSS is to protect cardholder data that is stored, processed or transmitted by merchants, banks, processors, gateway providers, point-of-sale vendors and hardware/software developers.

The most important step to ensuring SQL Server 2008 compliance is to make certain that your understanding of cardholder data and location of such data is clearly understood. “At a minimum, cardholder data contains the full Primary Account Number (PAN). Cardholder data may also appear in the form of the full PAN plus any of the following: Cardholder name, Expiration date, Service code.”¹

1.2 SQL Server Today

SQL Server has advanced over the years to become a very popular, capable database, evolving from a primarily departmental and SMB database to a fully enterprise capable platform. SQL Server’s appeals are many, from its highly scalable and secure database engine to its built in reporting and data analysis tools. SQL Server 2008 offers new features of particular interest to PCI DSS compliance including:

- Full Database Encryption through Transparent Data Encryption (TDE)
- Split Key Ownership through Extensible Key Management (EKM)
- Granular Auditing Capabilities through SQL Server Audit and Change Data Capture
- Continued support of Signed Module
- Built-in Control over Default SQL Server 2008 Features
- Stronger Control and Auditability over Server and Database Configuration through Policy-Based Management

¹ www.pcisecuritystandards.org/security_standards/pci_dss_supporting_docs.shtml

Implementation of the PCI DSS controls through SQL Server 2008 technology allows for the ability to standardize and computerize security controls effectively and efficiently, particularly when applied during the installation process.

1.3 PCI Requirements and SQL Server 2008

Six fundamental control objectives are what make up the core information security requirements for PCI DSS version 1.2. In an effort to encourage an information security best practice approach, twelve requirements were established in support of the control objectives. SQL Server 2008 features can be deployed to meet multiple PCI requirements. These requirements should always be considered by Information Technology personnel in cardholder environments when implementing SQL Server 2008.

The six control objectives and twelve requirements are as follows:

PCI Control Objective: Build and Maintain a Secure Network	
Requirement 1: Install and maintain a firewall configuration to protect cardholder data	N/A - Controlled at the network level
Requirement 2: Do not use vendor-supplied defaults	<ul style="list-style-type: none"> • SQL Server 2008 does not assign default passwords • Most SQL Server 2008 features are disabled by default • The sa account is disabled by default when SQL Server is setup using Windows Authentication • The BUILTIN/Administrators Windows role is not a member of the sysadmin group by default
PCI Control Objective: Protect Cardholder Data	
Requirement 3: Protect stored cardholder data	<ul style="list-style-type: none"> • SQL Server 2008 Transparent Data Encryption offers full data encryption • SQL Server 2008 cell level encryption offers encryption of individual columns • SQL Server 2008 Extensible Key Management offers split encryption key ownership
Requirement 4: Encrypt transmission of cardholder data across open, public networks	<ul style="list-style-type: none"> • SQL Server 2008 supports Secure Sockets Layer (SSL) encryption
PCI Control Objective: Maintain a Vulnerability Management Program	
Requirement 5: Use and regularly update	N/A - Controlled at the network level

anti-virus software	
Requirement 6: Develop and maintain secure systems and applications	Change controls are operational in nature; however, segregation of duties is addressed in requirement 7
PCI Control Objective: Implement Strong Access Control Measures	
Requirement 7: Restrict access to cardholder data by business need-to-know	<ul style="list-style-type: none"> • SQL Server 2008 Signed Module facilitates segregation of duties • SQL Server 2008 supports Windows Authentication • SQL Server 2008 supports Role Based Access
Requirement 8: Assign a unique ID to each person with computer access	<ul style="list-style-type: none"> • SQL Server 2008 supports Windows Authentication. Unique identification is preserved even when granted access as a member of a group. • The sa account is disabled by default when SQL Server is setup using Windows Authentication
Requirement 9: Restrict physical access to cardholder data	N/A - Physical access control
PCI Control Objective: Regularly Monitor and Test Networks	
Requirement 10: Track and monitor all access to network resources and cardholder data	<ul style="list-style-type: none"> • SQL Server Audit provides granular auditing capabilities • Once target systems are identified and PCI compliant configurations are set, SQL Policy-Based Management can track changes
Requirement 11: Regularly test security systems and processes	N/A - Controlled at the network level
PCI Control Objective: Maintain an Information Security Policy	
Requirement 12: Maintain a policy that addresses information security	N/A - Operational control procedures

2. SQL Server 2008 Features in Use

In the following sections, we will present an overview of the exciting new features of SQL Server 2008. This should aid many organizations to tactically and strategically comply with the PCI DSS. For each new feature that is relevant to the PCI DSS, we will present an overview of the feature, provide guidance on when to use that new feature and explain how the feature is enabled in many common scenarios. In some cases, your configuration requirements may be different than what we have presented, so you should consult your technical advisor. However, we believe these options should address most common scenarios.

2.1 Transparent Data Encryption (TDE)

Limiting sensitive cardholder data storage is critical to minimizing the risk of compromise. After you have addressed that simple tenet, SQL Server 2008 offers robust data encryption features which may help further reduce the risk of compromise or accidental disclosure. In most scenarios, where there is the requirement to store cardholder data, we have found that the TDE feature is an effective way to meet the PCI DSS for encryption of that data. TDE encrypts all database files including data files, log files and backup files. In some cases, continued use of cell level encryption is warranted.

Cell level encryption, introduced in 2005, continues to be a legitimate and useful feature in SQL Server 2008. Currently, many developers find that cell level encryption does not present a compelling cost/benefit since it requires modifications to client applications in order to handle the explicit encryption and decryption of data. It also prevents the encrypted cells from being indexed or sorted. In particular cases where other requirements dictate specific features not available from either TDE or cell level encryption, some organizations have found that third party add-on products are useful.

*"Transparent data encryption (TDE) performs real-time I/O encryption and decryption of the data and log files. The encryption uses a database encryption key (DEK), which is stored in the database boot record for availability during recovery. The DEK is a symmetric key secured by using a certificate stored in the master database of the server or an asymmetric key protected by an EKM module. TDE protects data "at rest", meaning the data and log files. It provides the ability to comply with many laws, regulations, and guidelines established in various industries. This enables software developers to encrypt data by using AES and 3DES encryption algorithms without changing existing applications."*²

When choosing to enable TDE in your environment there are a number of factors to consider during the implementation. First, TDE only secures data at rest and does not

² <http://msdn.microsoft.com/en-us/library/bb934049.aspx>

help to secure the communication (such as during remote ODBC queries) of the data. Second, the certificate used to encrypt the data is required during any attempt to decrypt the data. Third, complete and accurate backups of the certificate are required to minimize the risk of data loss. Backups of the database itself will be encrypted and will require the certificate as well.

2.1.1 Extensible Key Management (EKM)

Encryption keys may be protected manually using a database certificate or by utilizing a third-party encryption key management software package through SQL Server 2008's EKM. In many cases EKM software is the preferred solution, as it provides robust features for external key management and storage without requiring custom programming for many common tasks. Another consideration is your organization's experience with writing detailed security routines.

Using EKM requires knowledge of encryption key management best practices that many organizations choose to leave to specialists. For those that do not wish to invest in EKM software or custom programming, manual key management is available and permissible under the PCI DSS. Manual key management is considered by the authors of this paper to be more prone to errors and thus offers a lower level of real world protection.

EKM is typically used to protect keys used with TDE. To use TDE with EKM, changes are required to the underlying SQL Servers' configuration as well as to the SQL database instance where EKM/TDE will be used. EKM will need to be enabled on the SQL Server. Additionally, an EKM Provider will need to be installed and the SQL Server configured to use the new EKM Provider.

Now you are ready to create an asymmetric key in the master database with the EKM provider for use by TDE. At this point, your EKM software package will be able to take over to handle various key management tasks.

SQL Server 2008 also allows the use of a certificate, which has been generated using third party tools. An externally created certificate can be loaded instead of creating a new certificate manually. You should create the symmetric database encryption key protected by either the certificate or the EKM provided asymmetric key. This key encrypts the data itself. Please keep in mind that the database encryption key can only be created by SQL Server itself. Once the master key and certificates are created in the master database, the database can be configured to enable encryption.

2.1.2 PCI DSS Key Management Requirements

PCI DSS addresses the need for data encryption, as well as restricting and protecting access to encryption keys. Keys used for encryption/decryption of cardholder data must be secured against unauthorized use to reduce the risk of data exposure. This can be accomplished through implementation of split key management (e.g. dual control over keys), as well as through the rotation of keys on an annual basis.

Split key management is best handled through an EKM provider. An EKM provider can handle split key management by requiring multiple users to authenticate when performing administrative functions on the keys, such as changing permissions. To ensure segregation of duties, however, please bear in mind that the database owner and/or sysadmin should be independent of the EKM administrator. As another added control feature, EKM also separates the keys from the SQL Server application using the keys, so that keys are not stored with the data.

If you choose not to use an EKM provider, we suggest other ways to prevent a single user from having access to both the data and keys outside of the prescribed environment. First, any user that can backup keys and certificates should have write access to the backup folder location, but be denied read access to that location. Second, users with access to the key and certificate backup folders should be denied access to any backups of the database. To ensure that this is the case, the user who backs up the database should not be the same user who backs up the certificates.

If you are using manual key management several steps will be required. You will need to create a database master key in the master database (be sure to use a strong password to protect the key). The database master key you have created will be used to protect the TDE certificate. You are now ready to backup the master database master key to a removable disk and store in a safe location.

At this point, you are ready to create a certificate in the master database protected by the database master key. Once again, remember to backup the certificate to a removable disk and store in a safe location. Regardless of whether you are using an EKM provider or manually managing keys, remember that only users who need access to cardholder data should be given permissions to any keys and certificates used to decrypt sensitive data.

As noted above, encryption keys may be managed manually or through an encryption key management software package in SQL Server 2008. In the case of SQL Server, either the EKM generated asymmetric key or certificate that protects the TDE Database Encryption Key must be replaced at least once per year. You will need to generate or

load a new certificate or asymmetric key, backup the certificate, and re-encrypt the Database Encryption Key using the new certificate.

It is important to make sure to keep backups of prior certificates as those will be required to restore copies of the database made when those certificates were active. Keep in mind this is also required when using EKM generated asymmetric keys; however, the EKM provider should have features for managing this.

2.1.3 Other Encryption Considerations

The first way to protect cardholder information from being transmitted across open, public networks is to not send data across open, public networks. Obviously that is not entirely practical; however, SQL Server should never be directly accessible from the internet. The server should only be accessible through the context of a web application or web service or through a secure VPN.

Additionally, data transmitted from a client application of the SQL Server, such as an e-commerce web application, must be encrypted. The most secure SQL Server environment possible will fail if the applications attached to it are not secure. By default, SQL Server transmits data between the server and client unencrypted. SQL Server can be configured to encrypt that data using Secure Sockets Layer (SSL). Always encrypt sensitive cardholder information during transmission, even when data transmission is occurring within a corporate network.

2.2 SQL Server Audit

The audit functionality in SQL Server 2008 allows for granular control over what is logged. With this feature, actions, tables and users are auditable. In the event of a system compromise, auditing is critical in researching activity associated with a particular scenario. The ability to implicitly log and capture user access to cardholder data, detect changes to database objects/stored procedures, identify changes to server configuration settings and detect modifications to audit configuration settings (e.g. changes to audits and audit specifications) is key to achieving PCI compliance.

When considering what to audit, it is important to balance the need to determine what was accessed and/or manipulated against the amount of data being kept. On a database with high transaction volume, an audit log can rapidly grow to be many times the size of its source database. Keep in mind that an audit trail history must be retained for one year. The guidelines below include minimum data requirement for audit as per the PCI DSS.

First, we recommend that the following system activities be audited/logged:

- Login attempts - both successful and failed login attempts
- Server configuration - changes to encryption keys, creation, deletion or modification of logins and server level permissions, creation and deletion of databases
- Database - creation, deletion or modification of schema objects such as tables, stored procedures and views, addition or deletion of roles and users and changes to their permissions
- Data - auditing of any actions, inserts, deletions, updates or selects against tables containing cardholder data

Audits should be configured in such a way as to prevent tampering from SQL Server users, including members of the sysadmin fixed server role and from Windows users who may try to access the audit files without using SQL Server. To accomplish this, we suggest placing audit files in folders or file shares that are not accessible to members of the SQL Server sysadmin role and end users.

Additionally, we suggest giving the SQL Server service account only write access capability to the audit files folder and auditing actions against audits. Examples of actions to audit include changes to audit specifications and enabling or disabling of audits (SQL Server 2008 implicitly does this) and configuring audits to shut down the server if the audit fails. To do this, specify the ON_FAILURE = SHUTDOWN audit option when creating a server audit.

Another option is to write to the Windows Security event log. This allows for increased security over the log data but has a number of potential downsides:

- The security log contains all Windows security log events, not just those for SQL Server, thus making the detection of SQL Server issues more difficult
- Using the event log is slower than writing to a file and can affect server performance
- If the event log is set to overwrite when full, then a malicious user could purposefully fill the event log to cover their tracks
- Likewise, if the event log is set to not overwrite when full and it becomes full, it stops recording events. Unfortunately, SQL Server does not detect this and audit events will not be recorded.

It is recommended that, if using the Windows Security log to record SQL Server audit data, the Audit Collection Services of System Center Operations Manager 2007 be utilized to securely collect and store audit data outside of the log. Second, audit specifications should be defined on the server. These audit groups should be specified on SQL Server for any PCI compliant server and should apply to all users:

- SUCCESSFUL_LOGIN_GROUP
- LOGOUT_GROUP
- FAILED_LOGIN_GROUP
- LOGIN_CHANGE_PASSWORD_GROUP
- SERVER_ROLE_MEMBER_CHANGE_GROUP
- BACKUP_RESTORE_GROUP
- DBCC_GROUP
- SERVER_OPERATION_GROUP
- AUDIT_CHANGE_GROUP
- SERVER_STATE_CHANGE_GROUP
- SERVER_OBJECT_CHANGE_GROUP
- SERVER_PRINCIPAL_CHANGE_GROUP
- SERVER_PRINCIPAL_IMPERSONATION_GROUP
- SERVER_OBJECT_OWNERSHIP_CHANGE_GROUP
- SERVER_PERMISSION_CHANGE_GROUP
- SERVER_OBJECT_PERMISSION_CHANGE_GROUP

Third, database audit specifications should be defined. These audit groups should be applied to any PCI compliant databases and should apply to all users:

- APPLICATION_ROLE_CHANGE_PASSWORD_GROUP
- DATABASE_CHANGE_GROUP
- DATABASE_OWNERSHIP_CHANGE_GROUP
- DATABASE_PERMISSION_CHANGE_GROUP
- DATABASE_OBJECT_ACCESS_GROUP

Fourth, the below audit groups should be applied to any table in PCI compliant databases and should apply to all users:

- SELECT (Note that SELECT audits capture the SELECT statement and not the resulting data)
- INSERT
- UPDATE
- DELETE

Fifth, since the Database Engine can access the audit file, SQL Server logins with CONTROL SERVER permission can use the Database Engine to access the audit files. Define an audit on *master.sys.fn_get_audit_file* to record anyone reading the audit file. This will record which logins with CONTROL SERVER permission have accessed the audit file through SQL Server.

Although the above audit specifications will audit actions, such as inserts updates and deletes, actual changes to data are not audited. To do this, you should enable the Change Data Capture feature against any table containing cardholder data. Change Data Capture creates a change data capture table instance for each table being captured.

The Change Data Capture instance table contains all of the columns in the source table as well as five metadata columns to store information about the change. Unlike trigger based methods for capturing changes, Change Data Capture is implemented asynchronously using the log file and so have a far smaller effect on performance than trigger based methods.

Lastly, it is important to use Windows Authentication or at least an individual SQL Server login for each user as the audit functionality is dependent upon identifying the logged in user in the audit log. If a single application login or shared login is used then identifying a particular user making changes will be impossible. Using Windows authentication may make it easier to link and trace actions beyond the boundaries of the SQL Server.

2.3 Enhanced SQL Server 2008 Access Security Features

Protecting access to cardholder data through effective logical access controls is critical to ensuring PCI requirements are satisfied. Signed modules, introduced in 2005, allows for the facilitation of segregation of duties and continues to be supported in SQL Server 2008. With SQL Server 2008, the sa account is disabled by default when SQL Server is installed using Windows Authentication Mode, the BUILTIN/Administrators group is no longer a member of the sysadmin fixed server role and role based access is supported. These are valuable control features that are discussed in detail below.

2.3.1 Signed Module

A signed module is a view, stored procedure or function that is marked as signed by a certificate based login. The permissions of a certificate login used to sign the module are added to those of the executing user. If the signing login is a member of the sysadmin role, the module can perform any action on the server. Any authorized user of the module runs under that sysadmin context and is granted sysadmin permissions.

For example, through a signed stored procedure, a non-sysadmin user could be given the capability of creating new logins and users for the application that they administer without full sysadmin privileges. A good way to accomplish segregation of duties is to encapsulate required sysadmin functionality in signed modules and then grant non-sysadmin users access to those modules.

2.3.2 Windows Authentication

When establishing access management in SQL Server 2008, we recommend using Windows authentication unless client applications require mixed authentication. Additional items for consideration include assigning unique login ID's, avoiding the use of single application login ID's, disallowing users to share logins and creating individually mapped logins instead of creating logins for Windows groups that are members of the sysadmin role. The below login configurations should also be considered:

- Enforce password policy to enforce policy set by local or Active Directory (AD) security policy (this includes both SQL Server logins and, at the AD level, AD logins)
- Enforce 90 day password expiration (this includes both SQL Server logins and, at the AD level, AD logins)
- Set user must change password at next login
- Map only to databases needing access to
- Assign to database role(s) with the least privileges required
- Assume that the user will be able to by pass the application and connect directly to the SQL Server by other means. The user should have no more privileges connecting directly to SQL Server than through the application
- Consider creating an application role in the database to limit what a user can do when connecting directly to a database outside of the application
- SQL Server service accounts should be domain accounts with limited privileges

2.3.3 BUILTIN/Administrators Group

A critical step in restricting access to cardholder data is to limit the number of privileged users assigned to the sysadmin server role. By default, the BUILTIN/Administrators group is not a member of the sysadmin role in SQL Server 2008. The PCI DSS directly supports the best practice concept of "least privilege" access model therefore, we recommended the following:

- Members of the sysadmin role should only login using individual Windows Logins
- The number of users assigned to the sysadmin role should be minimal
- The sysadmin role should be assigned based on job function
- Members of the sysadmin role's Windows logins should be individually given access to SQL Server rather than through a mapped AD group
- Members of the sysadmin role should not be local Windows administrators
- Members of the sysadmin role should not have access to any folders or file shares on the server that the SQL Server service has access to such as, the folders containing the data and log files, and directories that databases or encryption keys could be backed up to
- A Windows local or domain administrator should not have sysadmin access to SQL Server

2.3.4 Role Based Access and Other Access Control Considerations

The following should also be considered when designing/managing access controls:

- The database owner should not be a sysadmin
- Where SQL Server authentication is allowed, the sa login should be disabled
- SQL Server users should have no local login, RDP or file access to the SQL Server machine
- Database security roles should be setup to restrict access to cardholder data
- Only those users with the need to see cardholder data should be placed in roles with access to cardholder data
- All users, including sysadmin members, should be prevented from being able to modify any audit log files
- Systems developers should not have the ability to make any database modifications to databases on the production server
- Changes to audit specifications should be logged (refer to section 2.2 for additional details)

2.4 Default SQL Server 2008 Features

One of the most basic ways to defend anything is to decrease its size. This concept is easily applied to a server to ensure that the area is less prone to attack. Reducing surface area on a server is essential to minimizing the risk of a system compromise. In SQL Server 2008 by default, most features, such as database mail and CLR integration, are disabled.

It is important to carefully choose only those items required by your application(s) to be installed or enabled and to monitor the configuration to ensure that unused features continue to be disabled. Each item that is added creates another potential point of attack. To ensure a secure installation, only the database engine itself should be installed. When required add functionality such as Reporting Services, Analysis Services, Integration Services or the Client Tools (see complete list below).

One service that may be enabled by default is the SQL Server Browser. If you are installing only the default instance of SQL Server, the SQL Server Browser service will be disabled. However, when you install a named instance of SQL Server, the SQL Server Browser service is enabled. Keep in mind that the SQL Server Browser service allows client applications to query for the dynamically allocated port of a named instance of

SQL Server. By configuring a fixed port for named instances of SQL Server, the SQL Server Browser service is not needed and may be disabled.

SQL Server services which are disabled by default:

- Database Mail
- SQL Mail
- CLR Integration
- OLE Automation
- XPCmdShell
- Ad Hoc Remote Queries
- Web Assistant
- Cross Database Ownership Chaining
- Service Broker
- SOAP

Another way to help increase security during installation or setup is to configure segregation of duties across domain accounts used for SQL Server 2008 and its related services. Configuring services to run using separate accounts prevents a successful compromise of one account leading to the compromise of unrelated data.

When not deploying SQL Server 2008 on Windows Server 2008 or Vista, it is important to properly configure the accounts that SQL Server 2008's various services use. To help ensure that a single compromised account does not impact the entire system do not use the SYSTEM NETWORK SERVICE or local administrator accounts for any service. In most cases, where these accounts are used, there is no technical basis for doing so.

When deploying SQL Server 2008 on Windows Server 2008 or Vista, the installation process uses the service security identifier (SID) feature of Windows to create unique service SIDs for each service, each of which have their own permissions. This has a similar effect of creating separate accounts for each service even when using the same service account.

Other steps that can be useful in securing the initial setup of the SQL Server include:

- Enable TCP/IP while disabling Named Pipes and VIA (Virtual Interface Architecture)
- On a default instance of SQL Server, change the port number to something other than the default value of 1433. Ensure that client connections can be configured to use a non-default port
- On a named instance of SQL Server, configure a specific port instead of allowing for dynamic ports

2.5 Policy-Based Management (PBM)

Prior to SQL Server 2008, monitoring the configuration of a SQL Server or database would have been a largely manual task, potentially assisted with custom scripts. SQL Server 2008 introduces the concept of PBM. PBM is designed to significantly change the way administrators manage the SQL Server data platform.

Today, database administrators (DBA) spend a large amount of time reacting to issues caused by configuration changes or deployments that do not comply with best practice “standards” or regulatory requirements. PBM allows the DBA to declare the desired state of the SQL Server environment and then manually or automatically check compliance of the system to that desired state.

The DBA declares intent as a Policy. Policy is the unit of automation/action, capturing the desired state (e.g. condition), where to apply (e.g. object set) and when to check (e.g. evaluation mode). The object set can be any entity in the instance, such as databases, tables, views and stored procedures. The evaluation mode can be check on schedule, check on change-log only, check on change-prevent or on demand.

A feature of PBM is that policies can be exported and imported into other servers and thus applied across an entire enterprise. In addition, policies can be applied to individual databases. Thus one set of policies can be applied only to PCI compliant databases while other databases may have more lenient policies applied. There may be a group of policies used to track and monitor access to cardholder data that makes sure there is an audit specification setup for changes to all stored procedures, views and functions.

If a policy evaluation is performed when an object changes, for instance an audit is turned off, it can either log the change or potentially prevent it (in some cases a DBA can decide to override or disable a preventing policy). If a test is performed on a scheduled basis or on demand it simply reports the failure. Target systems should be monitored and periodically validated to ensure that configured PCI relevant security policies are not modified.

2.5.1 Encryption Policy

To continually test and ensure that the settings referenced in section 2.1.1 are enabled, use SQL Server Policy Management to test for the following values:

Facet	Check Condition	Object Set	Notes
Database Facet	@EncryptionEnabled = True	PCI database(s)	
Server Configuration Facet	@ExtensibleKeyManagementEnabled = True	Server	Only if using EKM
Symmetric Key Facet	@EncryptionAlgorithm = TripleDes or AES256	Database Encryption Key	
Certificate Facet	@PrivateKeyEncryptionType = MasterKey	TDE certificate in the Master database	Only if not using EKM
	@KeyLength = 1024	Database Encryption Key	
	@ExpirationDate - @StartDate < 1 Year	TDE certificate in the Master database	Only if not using EKM
	@ExpirationDate > Today's Date	TDE certificate in the Master database	Only if not using EKM

2.5.2 SQL Server Audit Policy

To continually test and ensure that the settings referenced in section 2.2 are enabled, use SQL Server Policy Management to test for the following values:

Facet	Check Condition	Object Set	Notes
Server Facet	@AuditLevel = All	Server	
Audit Facet	@OnFailure = Shutdown	All audits	Be aware that specifying shutdown on failure can potentially cause reliability issues. In particular, avoid writing to network shares due to unreliable network writes.
	@Enabled = True	All audits	
Database Audit Specification	@Enabled = True	All database audit specifications	
Server Audit Specification	@Enabled = True	All server audit specifications	

2.5.3 Role Based Access Policy

To continually test and ensure that the settings referenced in section 2.3.4 are enabled, use SQL Server Policy Management to test for the following values:

Facet	Check Condition	Object Set
Database Security Facet	@IsOwnerSysadmin = False	PCI Database(s)

2.5.4 Authentication Policy

To continually test and ensure that the settings referenced in section 2.3.2 are enabled, use SQL Server Policy Management to test for the following values:

Facet	Check Condition	Object Set	Notes
Server Security Facet	@LoginMode = Integrated	Server	When using only Windows integrated authentication
Login Facet	@LoginType = WindowsUser	Logins that have access to the PCI database(s)	For Windows integrated logins
	@PasswordExpirationEnabled = True	All SQL logins	Only when SQL authentication is enabled.
	@PasswordPolicyEnforced = True	All SQL logins	Only when SQL authentication is enabled.
User Facet	@LoginType = WindowsUser	All users in PCI database(s)	

2.5.5 Factory Default Settings Policy

To continually test and ensure that the settings referenced in section 2.4 are enabled, use SQL Server Policy Management to test for the following values:

Facet	Check Condition	Object Set
Server Facet	@NamedPipesEnabled = False	Server
	@TcpEnabled = True	Server
Server Security Facet	@CrossDBOwnershipChainingEnabled = False	Server
Surface Area Configuration Facet	@AdHocRemoteQueriesEnabled = False	Server
	@ClrIntegrationEnabled = False	Server
	@DatabaseMailEnabled = False	Server
	@OleAutomationEnabled = False	Server
	@ServiceBrokerEndpointActive = False	Server
	@SoapEndpointsEnabled = False	Server
	@SqlMailEnabled = False	Server
	@WebAssistantEnabled = False	Server
	@XPcmdShellEnabled = False	Server
	@ClrIntegrationEnabled = False	Server

3. Conclusion

Implementing Microsoft SQL Server 2008 requires careful planning, analysis and an understanding of the impact to interfaces based on other technology in the environment. System developers and IT leadership can no longer ignore security, privacy and regulatory requirements that mandate compliance with the storage, transmitting and processing of data. This paper highlighted key areas for a developer to focus on to confirm cardholder data is appropriately protected.

Data encryption, dual controls, effective audit trails, accountability, strong password controls, implementation of roles based access and audits of system configuration changes are core security components that are critical to protecting data. Automated implementation of such controls in SQL Server 2008 would allow for the ability to achieve PCI compliance as well as standardize and computerize security controls effectively and efficiently. In order to ensure risk is mitigated in the environment, we encourage developers to continually assess their environments, stay abreast of requirements in the industry and consider the ramifications of not being compliant.

This white paper provided an overview of configurable PCI DSS requirements and proposed solutions, however, the following resources in section 3.1 are provided below for additional reference. We recommend consulting a Qualified Security Assessor (QSA) to evaluate specific configuration issues related to the PCI standards.

3.1 Resources

For more information on implementing SQL Server 2008 based on the PCI Data Security Standards, please visit the following sites:

- PCI Security Standards Council website: <https://www.pcisecuritystandards.org>
- PCI Data Security Standard version 1.2:
http://www.pcisecuritystandards.org/tech/download_the_pci_dss_htm
- PCI Glossary:
http://www.pcisecuritystandards.org/security_standards/pci_dss_supporting_docs.shtml
- Understanding Transparent Data Encryption (TDE):
<http://msdn.microsoft.com/en-us/library/bb934049.aspx>
- Understanding Extensible Key Management (EKM):
<http://msdn.microsoft.com/en-us/library/bb895340.aspx>
- Understanding SQL Server Audit: <http://msdn.microsoft.com/en-us/library/cc280386.aspx>
- SQL Server Audit Action Groups and Actions: <http://msdn.microsoft.com/en-us/library/cc280663.aspx>
- Change Data Capture: <http://msdn.microsoft.com/en-us/library/bb522489.aspx>

- Module Signing (Database Engine): <http://msdn.microsoft.com/en-us/library/ms345102.aspx>
- How to Configure a Server to Listen on a Specific TCP Port (SQL Server Configuration Manager): <http://msdn.microsoft.com/en-us/library/ms177440.aspx>
- How to Enable or Disable a Server Network Protocol (SQL Server Configuration Manager): <http://msdn.microsoft.com/en-us/library/ms191294.aspx>
- Understanding Surface Area Configuration: <http://msdn.microsoft.com/en-us/library/ms161956.aspx>
- Administering Servers by Using Policy-Based Management: <http://msdn.microsoft.com/en-us/library/bb510667.aspx>
- Encryption Hierarchy: <http://msdn.microsoft.com/en-us/library/ms189586.aspx>