# Общее описание

Публичное API ПО «Хранитель 2016» (далее АПИ) предназначено для обмена информацией с внешними системами. Под внешними системами в понимание этого документа понимаются – программное обеспечение, расположенное исключительно внутри периметра предприятия, использующего ПО «Хранитель» и за пределами периметра ПО «Хранитель»

Обмен информации происходит с использованием файлов в формате xml, структура и наполнение которых должна соответствовать xsd-схеме, которая является составляющей этого документа.

Все документы которыми обмениваются ПО отправляются зашифрованными. Используется спецификация XML Encryption

На все документы которыми обмениваются системы накладывается цифровая подпись. Используется спецификация XML Signature

В качестве модулей наложения проверки ЭЦП. Могут использоваться модули сторонних разработчиков. Взаимодействие модулей ЦП с АПИ выполняется через БД АПИ. Описание интерфейса приведено в Приложении №

Внешние системы взаимодействуют с АПИ исключительно через коннекторы. Коннекторы в свою очередь взаимодействуют с АПИ в асинхронном режиме, с соблюдением принципа FIFO.

АПИ не инициирует соединение с внешними системами, но соединение с внешними системами могу инициировать коннекторы. Смотрите описание коннекторов

Коннекторы могут разрабатываться сторонними разработчиками и должны взаимодействовать с очередью сообщений API через соединение с БД. Описание смотрите в соответствующем разделе

Информация, участвующая в обмене, носит статус конфиденциальной, если не установлено другое, внутренними документами предприятия. Предоставление доступа к АПИ системам вне периметра предприятия нарушает безопасность ПО «Хранитель». Ответственность за представление такого доступа лежит исключительно на ответственных лицах предприятия использующего ПО «Хранитель»

# Обобщённая архитектурная схема.

Смотрите схему ниже

Элементы схемы:

Коннекторы - внешние службы задача которых состоит:

1. В получении от внешней системы пакета заданного формата и записи его в очередь.
2. Считывание из очереди сообщение и отправка или предоставление доступа к нему внешней системе

Модуль sign/crypt – предназначен для наложения/проверки ЭЦП на документ, шифровании и расшифровке тела сообщения

Custodian 2016 Public API Queue - очередь сообщений представляет собой таблицу в БД, как выделенной так и в БД ПО «Хранитель». Очередь содержит входящие и исходящие пакеты. При использовании коннекторов и/или модуля sign/crypt разработанных сторонними организациями, запрещается размещать объекты связанные с Custodian 2016 Public API Queue в БД ПО «Хранитель 2016»

Custodian 2016 Public API processing module – основная часть АПИ. Содержит в себе следующие функции:

* Обработка полученных пакетов от внешних систем
* Выполнение необходимых действий, определённых процедурой обработки каждого пакета
* Выполнение функций аутентификации приложения, сформировавшего пакет, на основании идентификатора приложения и результата проверки ЦП



Figure 1Обобщенная архитектурная схема

# Структура Custodian Public API processing module

Модуль администрирования

Представляет собой модуль в интерфейсе ПО «Хранитель 2016».

Задача:

* Добавление карточек внешних приложений
* Назначение доступа внешним приложениям в определённому набору запросов
* Установка сертификата приложения, на основании которого будут выполняться функции sign/crypt
* Установка временных регламентов выполнения транзакций

Модуль обработки

Задача данного модуля конвертация полученных сообщений в стандартный бизнес-процесс ПО «Хранитель»

# Типы запросов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Запрос | | Результат | |
| RequestType | Xsd класс | RequestType | Xsd класс |
| **Административные операции** | | | | |
| Создание карточки клиента, распорядителя, управляющего, брокера | NewCustomer | cust:TCustomer | Customer | cust:TCustomer |
| Обновление/изменение карточки клиента, распорядителя, управляющего, брокера | UpdateCustomer | cust:TCustomer | Customer | cust:TCustomer |
| Открытие счета в ЦБ | NewAccount | cust:TnewAccountRequest | Account | cust:TAccount |
| Обновление анкеты счета в ЦБ | UpdateAccount | cust:TAccount | Account | cust:TAccount |
| Поиск существующей карточки клиента | SearchCustomer | cust:TSearchCustomer | CustomerList | cust:CustomerList |
| Поиск существующей карточки счета в ЦБ | SearchAccount | cust:TSearchAccount | AccountList | cust:TAccountList |
| **Информационные** | | | | |
| Выписка о состоянии счета | Statement\_of\_Holdings\_Request | Cust:TStatement\_of\_Holdings\_Request | Statement\_of\_Holdings | cust:TStatement\_of\_Holdings |
| Выписка про операции | Statement\_of\_Transactions\_Request | cust:TStatement\_of\_Transactions\_Request | Statement\_of\_Transactions | cust:TStatement\_of\_Transactions |
| **Учетные** | | | | |
| Состояние счета в ЦБ | Balance | cust:TBalance\_request | Balance | cust:TBalance\_responce |
| Проверка возможности выполнения операции | Transaction\_pre\_check | cust:TTransaction\_pre\_check\_request | Transaction\_pre\_check | cust:TTransaction\_pre\_check\_responce |
| Транзакция | Transaction | Cust:TTransaction\_request | Transaction | Cust:TTransaction\_responce |
| Подтверждение транзакции клиентом | TransactionCommit | Cust:TTransactionCommit | Status | Cust:TStatus |
| Отказ клиентом от транзакции | TransactionRollback | Cust:TTransactionRollback | Status | Cust:TStatus |
| **Служебные** | | | | |
| Статусное сообщение |  |  | Status | Cust:TStatus |
| Сообщение об ошибке |  |  | Error | Cust:TStatus |



Figure 2 Общая схема обмена сообщениями

# Транзакции по ЦБ

Технологическая карта выполнения транзакций приведена ниже.

Статусы операций, которые используются в схеме

|  |  |
| --- | --- |
| Код | Название |
| 0 | Зареєстровано в журналі розпоряджень |
| 1024 | Виконано |
| 2048 | операція скасована |
| 3004 | операція скасована |
| 7000 | ЗС. Отримано від зовнішньої системи |
| 7001 | ЗС. Відмова. Недостатня кількість ЦП на рахунку |
| 7002 | ЗС. Чекає підтвердження. Недостатня кількість ЦП на рахунку |
| 7003 | ЗС. Відмова. Перевищено ліміт |
| 7004 | ЗС. Чекає підтвердження. |
| 7005 | ЗС. Очікуємо відкриття операційної доби |
| 7006 | ЗС. Відмова клієнта |



Figure 3Технологическая карта выполнения транзакции

# Шифрование и подпись сообщений.

Для подписи и шифрования сообщений используются спецификация XML Signature и XML Encryption.

Для каждого сообщения, отправляемого в системе:

Шаг 1. Наложение подписи на весь документ

Шаг 2. Шифрование раздела cust:body

При получении сообщения:

Шаг 1. Расшифровка сообщения

Шаг 2. Проверка подписи сообщения

Примеры функций подписи/шифрования

Подпись:

public static void Sign(XmlDocument doc, X509Certificate2 Cert)

{

if (!Cert.HasPrivateKey)

{

throw new Exception("Specified certificate not suitable for signing!");

}

XmlDsigExcC14NTransform transform = new XmlDsigExcC14NTransform();

SignedXml signed = new SignedXml(doc);

signed.SigningKey = Cert.GetRSAPrivateKey();

signed.SignedInfo.SignatureMethod = RSAPKCS1SHA256SignatureDescription.SignatureMethod;

signed.SignedInfo.CanonicalizationMethod = transform.Algorithm;

Reference reference = new Reference();

reference.Uri = "";

XmlDsigEnvelopedSignatureTransform env = new XmlDsigEnvelopedSignatureTransform();

reference.AddTransform(env);

signed.AddReference(reference);

signed.KeyInfo = CreateKeyInfoFromCertificate(Cert);

signed.ComputeSignature();

XmlElement xmlDigitalSignature = signed.GetXml();

doc.DocumentElement.AppendChild(doc.ImportNode(xmlDigitalSignature, true));

}

Вспомогательный метод для подписи

private static KeyInfo CreateKeyInfoFromCertificate(X509Certificate2 certificate)

{

KeyInfoX509Data kiData = new KeyInfoX509Data(certificate);

kiData.AddSubjectName(certificate.Subject);

KeyInfo keyInfo = new KeyInfo();

keyInfo.AddClause(kiData);

return keyInfo;

}

Проверка подписи

public static Boolean VerifyXml(XmlDocument xmlDoc, X509Certificate2 Cert)

{

if (xmlDoc == null)

throw new ArgumentException("xmlDoc");

SignedXml signedXml = new SignedXml(xmlDoc);

XmlNodeList nodeList = xmlDoc.GetElementsByTagName("Signature");

if (nodeList.Count <= 0)

{

throw new CryptographicException("Verification failed: No Signature was found in the document.");

}

if (nodeList.Count >= 2)

{

throw new CryptographicException("Verification failed: More that one signature was found for the document.");

}

signedXml.LoadXml((XmlElement)nodeList[0]);

XmlElement root = xmlDoc.DocumentElement;

return signedXml.CheckSignature(Cert,true);

}

Шифрование

public static void Encrypt(XmlDocument Doc, X509Certificate2 Cert)

{

string ElementToEncrypt = "cust:body";

if (Doc == null)

throw new ArgumentNullException("Doc");

if (Cert == null)

throw new ArgumentNullException("Cert");

XmlElement elementToEncrypt = Doc.GetElementsByTagName(ElementToEncrypt)[0] as XmlElement;

// Throw an XmlException if the element was not found.

if (elementToEncrypt == null)

{

throw new XmlException("The specified element was not found");

}

EncryptedXml eXml = new EncryptedXml();

EncryptedData edElement = eXml.Encrypt(elementToEncrypt, Cert);

EncryptedXml.ReplaceElement(elementToEncrypt, edElement, false);

}

Расшифровка

public static void Decrypt(XmlDocument Doc)

{

if (Doc == null)

throw new ArgumentNullException("Doc");

EncryptedXml exml = new EncryptedXml(Doc);

exml.DecryptDocument();

}