OstCard Data Processing Service

Руководство пользователя
Версия 1.0

Оглавление

Data Processing Service – программный продукт, предназначенный для конвертации данных из одного формата (например, XML) в другой (например, табличный) и наоборот. В общем случае программа применяется для обработки Request-файлов, выгруженных из системы OpenWay, и подготовки ответных Response-файлов в формате XML. В процессе обработки возможен обмен информацией с внешними источниками (например, сервер ЦИТ) и вызов сторонних программ подготовки данных (например, CDP).

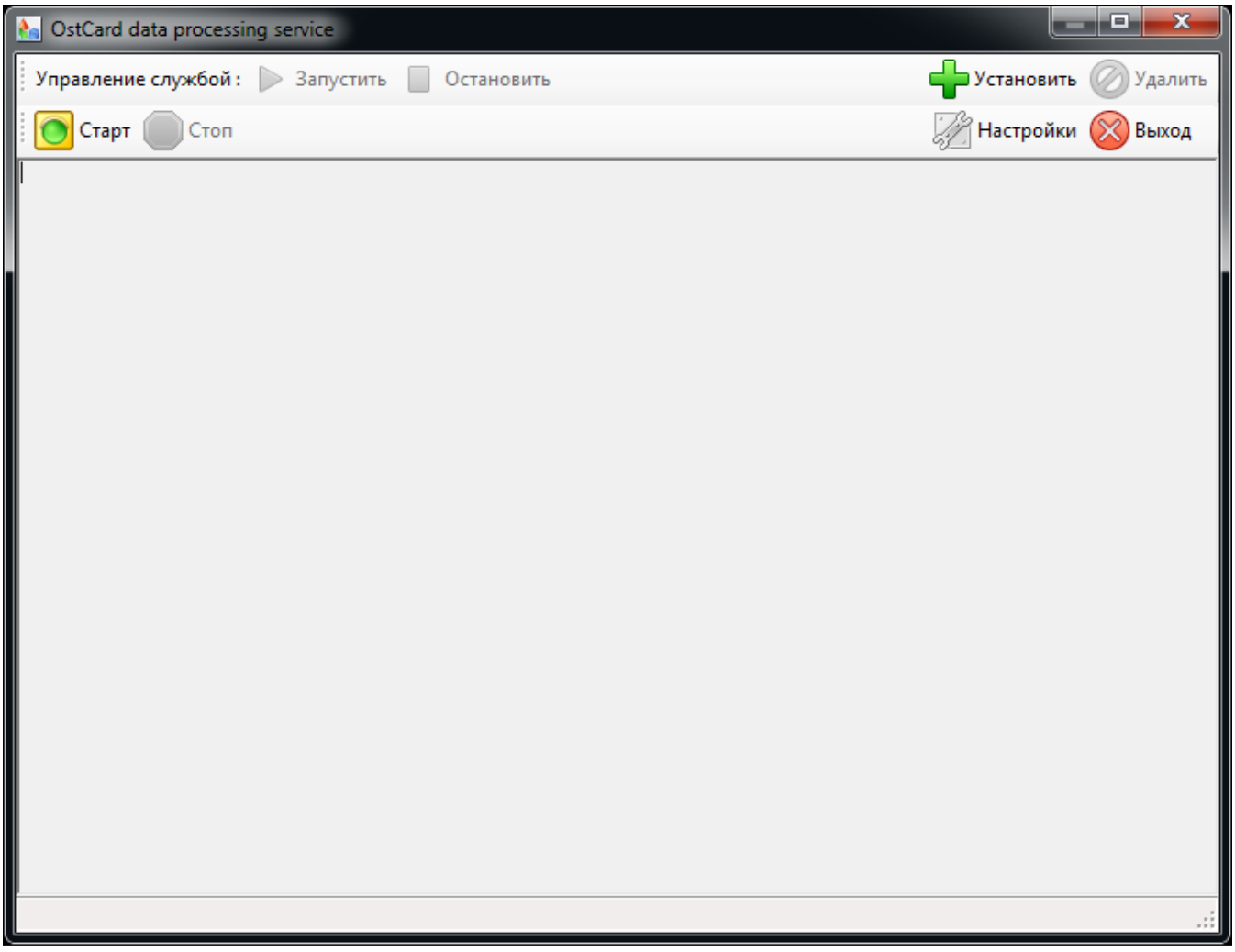
1. Инсталляция

Запустить SetupDataProcessingService.msi, далее следовать инструкциям. Пункт 'Установить службу Windows' можно не отмечать, службу можно будет установить из программы. Если пункт отмечен, служба будет инсталлирована, но не запущена.

Для работы программы требуется Microsoft .NET Framework 4.0. Если отсутствует, будет установлен автоматически при инсталляции.

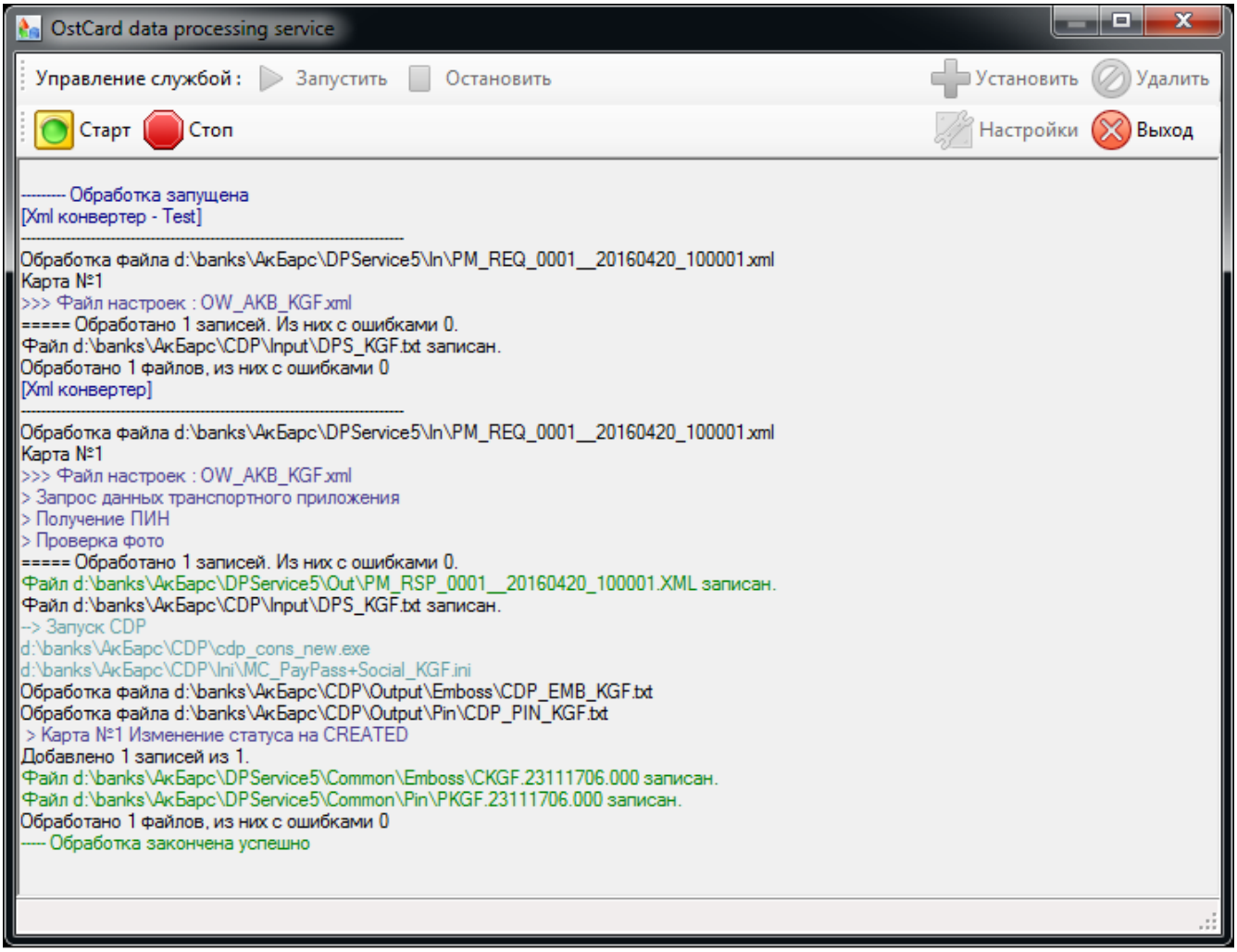
2. Запуск

Запустить программу 'OstCard Data Processing Service' можно с иконки на рабочем столе или из меню программ. Программа по умолчанию запускается в свернутом виде, в трее появится иконка. Двойное нажатие мышкой на иконку или пункт меню 'открыть' по правой клавише мыши открывает панель управления программой.



Верхняя панель инструментов управляет службой. Если служба установлена, программа управляет службой и выводит результаты работы службы. Для запуска службы нажать кнопку 'Запустить'. Если служба не установлена, программа работает как обычный исполняемый модуль Windows. В этом случае для начала работы следует нажать кнопку 'Старт'.

Результат работы программы:

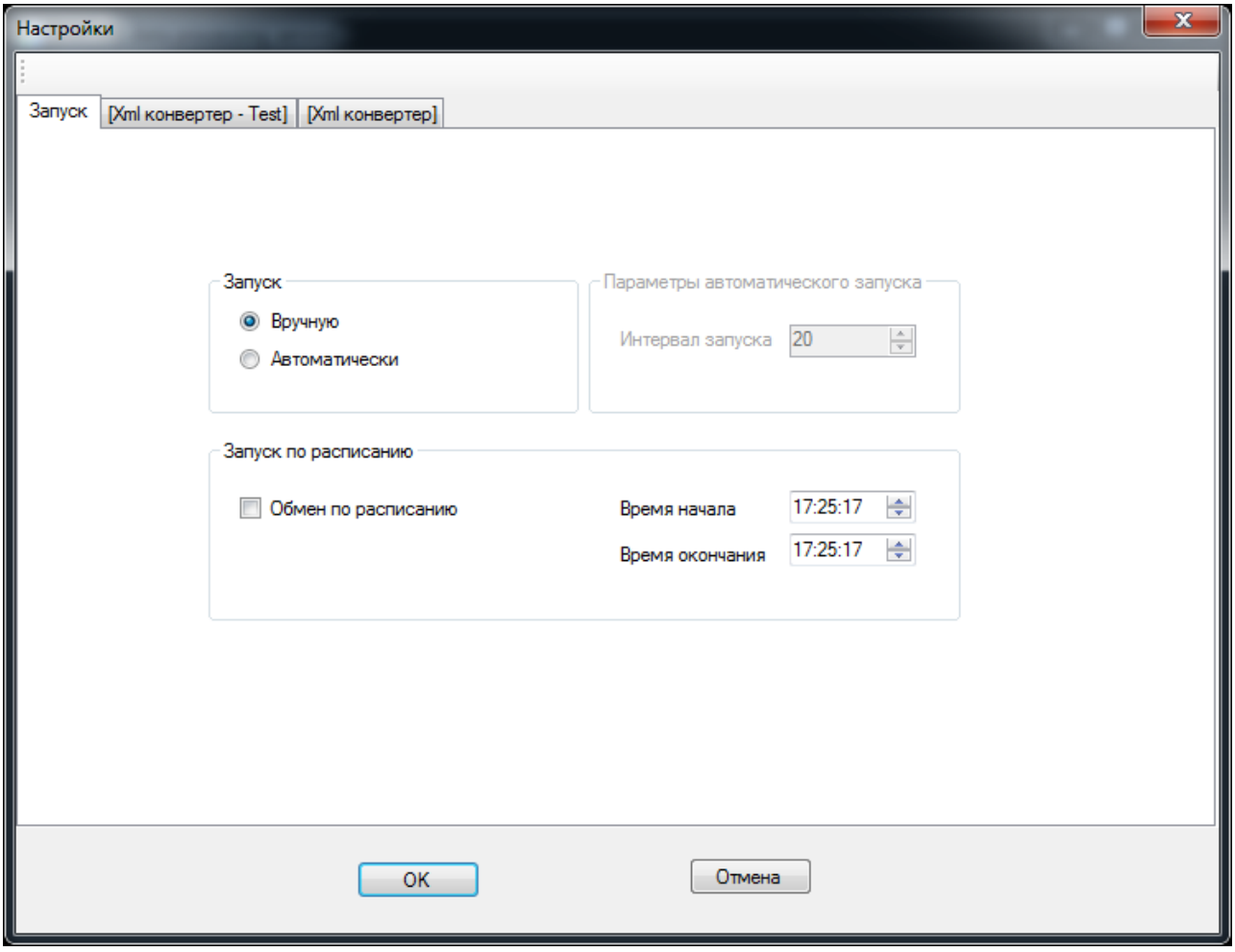


После начала работы программы нажатие кнопки 'Выход' приводит к сворачиванию окна. Чтобы завершить работу, надо нажать 'Стоп', потом 'Выход', либо 'Выход' по нажатию правой клавишей мыши на иконке в трее.

3. Кнопка 'Настройки'

Нажатие кнопки 'Настройки' открывает визуальный редактор настроек программы, который позволяет редактировать некоторые параметры обработчика (реализовано не полностью).

На вкладке 'Запуск' можно задавать режим запуска обработки (вручную или автоматически) и периодичность запуска (в секундах). Запуск по расписанию позволяет установить время начала и время окончания автоматических запусков программы (например, с начала и до конца рабочего дня).

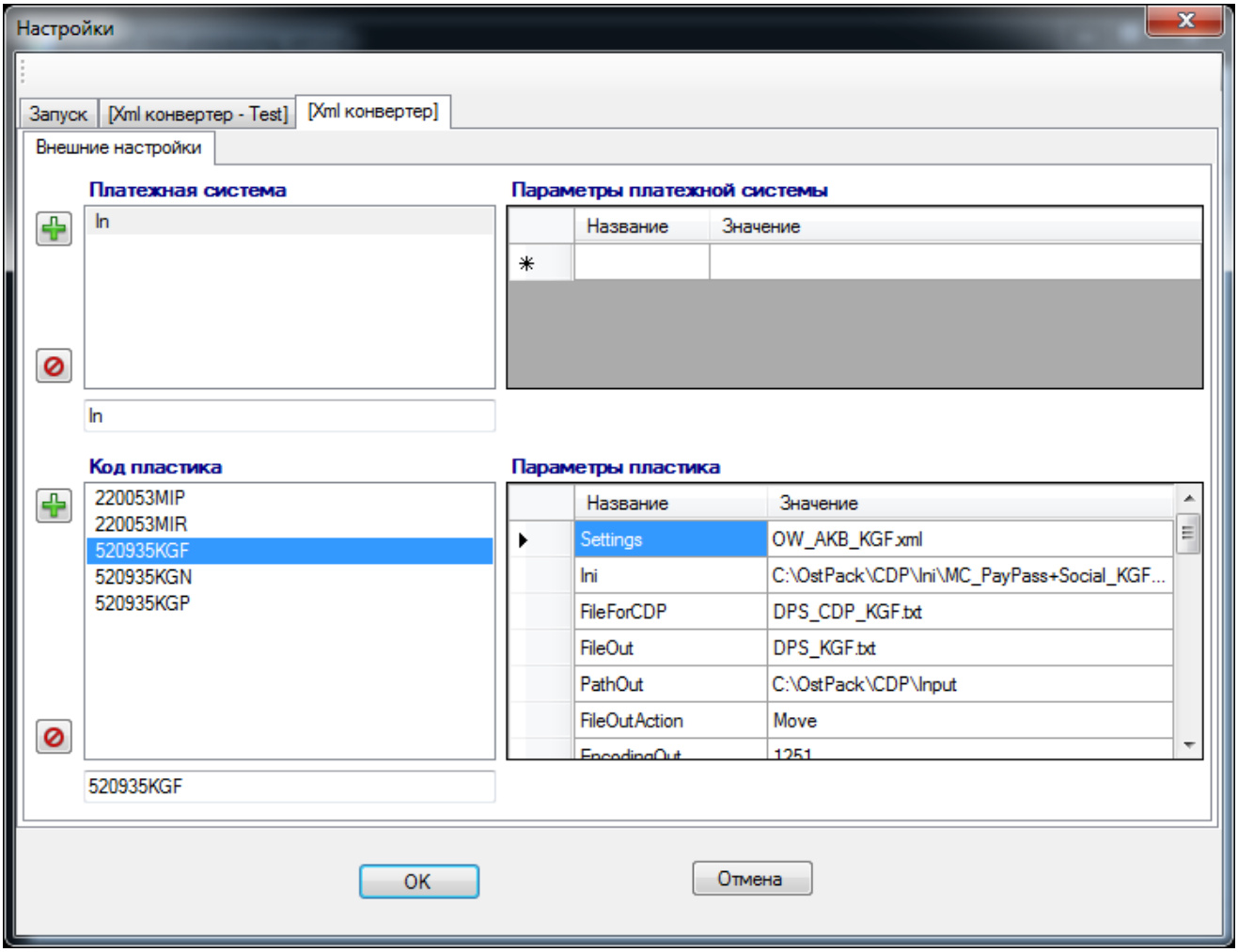


На вкладке 'Xml конвертер' определяются параметры обработки карточных продуктов. Каждый карточный продукт идентифицируется уникальным кодом пластика, который формируется из полей входного Request-файла (например, первые 6 цифр PAN и значение поля ADD_INFO_1).

'Платежная система' позволяет объединить карточные продукты в группы (например, по банкам или филиалам).

Кнопка 'Плюс' создает новую платежную систему или код пластика как копию текущего. Название присваивается в поле редактирования. Кнопка 'Удалить' удаляет текущий элемент.

Подробнее по конкретным параметрам кода пластика читайте в разделе 'Настройки'.



На вкладке 'Xml конвертер - Test' приведена копия настроек 'Xml конвертер'. Фактически 'Xml конвертер - Test' – это настройки тестового модуля, который выполняет предварительную обработку (или проверку) входных Request-файлов и находит грубые ошибки (такие как, например, отсутствие обязательных полей в Request-файле).

4. Настройки

4.1. Основные настройки (DPService.xml)

4.1.1. Настройки автоматической обработки (вкладка 'Запуск')

```
<SettingsRun>
 <!-- Обмен запускается автоматически или вручную. (false, true) -->
 <AutomaticRun>false</AutomaticRun>
 <!-- Запуск обмена по расписанию (время дня). (false, true) -->
 <ScheduleRun>false</ScheduleRun>
 <!-- Время начала обмена по расписанию. -->
 <StartTime>10:00:00</StartTime>
 <!-- Время окончания обмена по расписанию. -->
 <EndTime>22:00:00</EndTime>
 <!-- Интервал запуска обмена ( в секундах ). -->
 <Interval>20</Interval>
</SettingsRun>
```

4.1.2. Запускаемый модуль

Запускаемый модуль определяется в элементе <Module>. Таких элементов может быть несколько, программа будет выполнять их по очереди. Состав элемента :

<Title>[Xml конвертер]</Title> – заголовок, который выводится на экран при запуске и на вкладке настроек

<Dll>DP_OWUConverter.dll</Dll> – имя библиотеки dll, где реализованы алгоритмы обработки

<Class>DP_OWUConverter.OWUConverter</Class> – имя используемого класса

<Settings> – настройки для работы модуля

4.1.3. Настройки запускаемого модуля

Настройки для работы модуля находятся внутри элемента <Settings>. Далее все xml -пути будут определяться от этого элемента. Примерный набор настроек:

```
<WorkingDirectories>
     <Dir>
       <Path>I:\OW_Data\OSTCARD_WAY\IN_REQ</Path>
       <PaymentSystem>In</PaymentSystem>
     </Dir>
     <Dir>
       <Path>C:\OstPack\DpService\In</Path>
       <PaymentSystem>In</PaymentSystem>
     </Dir>
</WorkingDirectories>
```

<WorkingDirectories> – содержит один или несколько элементов <Dir>, описывающих каталоги для входных файлов в элементе <Path> и ключевые значения, привязанные к ним. В данном примере входные каталоги привязаны к 'Платежной системе' In.

В элементе <Common> задаются общие настройки обработчика. Каждая из этих настроек может быть переопределена в файле внешних настроек ExternalData.xml для каждого конкретного карточного продукта (см. описание в след. разделе). В случае переопределения будет применяться настройка из файла ExternalData.xml. Настройки в файле DPService.xml применяются по умолчанию.

```
<Common>
     <PathSettings>C:\OstPack\DPService\Settings</PathSettings>
     <PathOut>C:\OstPack\CDP\Input</PathOut>
     <InputFileAction>Move</InputFileAction>
     <UnionInputFiles>false</UnionInputFiles>
     <PathPersoData>c:\OstPack\DPService\PersoData</PathPersoData>
     <SearchPattern>*.xml</SearchPattern>
     <ExitError>false</ExitError>
     <XmlError>true</XmlError>
     <!-- Test,Txt,TxtXml-->
     <ProcessingType>TxtXml</ProcessingType>
     <!-- None,NonCritical,Input,Critical -->
     <DefaultErrorType>Input</DefaultErrorType>
     <PathXMLTemplates>C:\OstPack\DpService\Templates</PathXMLTemplates>
     <PathRsp>C:\OstPack\DpService\Out</PathRsp>
     <!-- может не быть, по умолчанию - true -->
     <RandomPIN>true</RandomPIN>
     <PathTerritoryData>C:\OstPack\DpService\TerritoryData</PathTerritoryData>
     <ReplaceChars>",'</ReplaceChars>
     <TIMEOUT>60</TIMEOUT>
     <!-- может не быть, по умолчанию - true -->
     <DetailLog>true</DetailLog>
     <!-- может не быть, по умолчанию - false -->
     <ValidatePassport>false</ValidatePassport>
     <CommonDelimiter>___</CommonDelimiter>
```

```
</Common>
```

<PathSettings> – каталог, где находятся файлы настроек (подробнее о файлах настроек см. ниже)
<PathOut> – каталог, куда будут записываться входные файлы для CDP
<InputFileAction> – действие со входным Request-файлом после его обработки. Возможные значения:

None - оставить
Move - переместить в каталог 'Save' при успехе, в каталог 'Error' при ошибке
MoveError - переместить в каталог 'Error' при ошибке, при успехе оставить
MoveOK - переместить в каталог 'Save' при успехе, при ошибке оставить
Delete - удалить в любом случае
DeleteOKMoveError - удалить при успехе, при ошибке переместить в каталог 'Error'
DeleteOKNoneError - удалить при успехе, при ошибке оставить
DeleteErrorMoveOK - удалить при ошибке, при успехе переместить в каталог 'Save'
DeleteErrorNoneOK - удалить при ошибке, при успехе оставить

<UnionInputFiles> – объединять ли при обработке входные Request-файлы (true) или обрабатывать их по одному (false)
<PathPersoData> – каталог, где размещаются файлы, содержащие реестр с информацией о гражданах
<SearchPattern> – шаблон выбора входных Request-файлов (этого элемента может не быть, по умолчанию выбираются все файлы с расширением xml из входного каталога)
<ExitError> – прекращать ли обработку в случае возникновения ошибки
<XmlError> – формировать ли дубликат входного Request-файла в папке XmlError с картами, по которым возникла ошибка в процессе обработки
<ProcessingType> – тип обработчика. Возможные значения:

Test - тестовый (выполняет только проверку входных Request-файлов и настроек обработчика на наличие грубых ошибок, не связывается с ЦИТ и не формирует никаких выходных файлов)
Txt - формирует только входной файл для CDP и выходные Common-файлы без Response-файлов
TxtXml - формирует входной файл для CDP, выходные Common-файлы и Response-файлы

<DefaultErrorType> – тип ошибки по умолчанию. Этот тип ошибки будет применяться к полям, для которых явно не задан тип ошибки в атрибуте ErrorType. Возможные значения:

None - все возможные ошибки в полях будут игнорироваться, обработка прерываться не будет
NonCritical - некритичная ошибка, обработка карты с такой ошибкой будет отложена и информация о ней переместится во временный файл в каталоге XmlError (при этом в выходной Response-файл такая карта попадет)
Input - ошибка входных данных, применяется по умолчанию ко всем полям, которые берутся из входного Request-файла (возникает когда, например, обязательное поле отсутствует в Request-файле; в этом случае обработка прервется и такой Request-файл переместится в каталог Error)
Critical - критическая ошибка (возникает когда, например, не удается сгенерировать ПИН-блок для карты; в этом случае обработка прервется и Request-файл переместится в каталог Error)

<PathXMLTemplates> – каталог, где находятся шаблоны выходных Response-файлов
<PathRsp> – каталог, где формируются выходные Response-файлы
<RandomPIN> – генерировать ли для каждой карты случайный ПИН (true) или использовать константу (false)
<PathTerritoryData> – каталог, где размещаются файлы, содержащие описание кодов территорий
<ReplaceChars> – заменять в значениях полей один символ на другой (в примере ",' означает замену " на ')
<TIMEOUT> – таймаут на соединение с ЦИТ в секундах
<DetailLog> – вести детальный лог обработки каждого входного Request-файла в каталоге Detail
<ValidatePassport> – проверять ли валидность паспорта гражданина в ответе ЦИТ
<CommonDelimiter> – окончание, которое добавляется к названию Common-файлов в случае настройки их разделения по количеству карт (см. описание настроек ExternalData.xml ниже)

Элемент <ProcessingFormats> содержит один или несколько элементов <Format>, описывающих структуру входных xml-файлов для обработки:

```
  <ProcessingFormats>
       <Format>
         <Namespaces>
              <Namespace>
                <!-- Пространство имен, определенное в атрибуте 'xmlns' -->
                <Uri>http://namespaces.globalplatform.org/systems-messaging/1.0.0</Uri>
                <!-- Строка, заменяющая пространство имен, для удобства написания -->
                <Prefix>gp</Prefix>
              </Namespace>
         </Namespaces>
<XPathParent>PMJobs/gp:GPMessage/gp:GPBody/gp:ApplicationDataNotification</XPathParent>
         <FindData>
              <Bank Type="Text">FileHeader/Destination</Bank>
              <ComplexData Dynamic="true" Replaced="true">
                <Key>PLSC</Key>
                <Field>
                     <Name>CardProduct</Name>
                     <Field Type="Attribute" Name="Value">
                       <Name>PAND</Name>
<XPath>gp:ApplicationDataPerCRN/gp:ApplicationData/gp:ICCData/gp:DataSet[@Name='ADTA']/gp:Dat
a[@DataElement='PAND']</XPath>
                       <Required>true</Required>
                       <SubStr Index="0" Length="6" />
                     </Field>
                     <Field Type="Attribute" Name="Value">
                       <Name>ADD_INFO_1</Name>
<XPath>gp:ApplicationDataPerCRN/gp:ApplicationData/gp:ICCData/gp:DataSet[@Name='ADTA']/gp:Dat
a[@DataElement='ADD_INFO_1']</XPath>
                       <Required>true</Required>
                     </Field>
                </Field>
              </ComplexData>
         </FindData>
         <ExternalFile>
              <File>C:\OstPack\DPService\ExternalData.xml</File>
              <SettingsPath>PaymentSystem/PLSC/Settings</SettingsPath>
         </ExternalFile>
       </Format>
  </ProcessingFormats>
```

<XPathParent> – задает корневой путь до данных карты во входном Request-файле, относительно которого будут задаваться все остальные пути до полей с данными
<FindData> – определяет формирование кода пластика для идентификации карточного продукта для каждой карты во входном Request-файле

<ComplexData> – определяет формирование составного поля с ключом <Key>PLSC</Key> как объединение двух полей из входного Request-файла: **первые 6 цифр PAN карты из поля PAND и значение поля ADD_INFO_1**
<ExternalFile> – определяет путь до файла внешних настроек ExternalData.xml в элементе <File>; в элементе <SettingsPath> задается путь до элемента в ExternalData.xml, содержащего название файла настроек: PaymentSystem/PLSC/Settings (здесь ключ PaymentSystem определяется по входному каталогу, PLSC определяется при формировании кода пластика, значение элемента Settings берется из ExternalData.xml)

В элементе < AdditionalActions> указывается название библиотеки dll, содержащей специфичные функции обработки:

```
<AdditionalActions>
     <Dll>AKBActions.dll</Dll>
     <Class>AKBActions.Actions</Class>
</AdditionalActions>
```

В элементе Actions/CDP/FileExe прописывается путь до консольной программы CDP:

```
<Actions>
     <Action Name="BeforeOpenRoot" />
     <Action Name="AfterProcess">
       <CDP>
     <FileExe>C:\OstPack\CDP\cdp_cn.exe</FileExe>
       </CDP>
     </Action>
</Actions>
```

4.1.4. Логика обработки входных файлов

Последовательно просматриваются все элементы <Dir> внутри элемента <WorkingDirectories> (WorkingDirectories/Dir) . Внутри элемента <Dir> должен быть обязательный элемент <Path>, задающий каталог входных файлов. Если путь относительный, он строится от каталога, куда была инсталлирована программа. Остальные элементы, которые могут быть внутри элемента <Dir>, определяют пару ключ(название элемента)/значение(текстовое значение элемента) для поиска дополнительных данных в файле внешних настроек ExternalData.xml (определен в Format/ExternalFile/File, подробнее см. ниже). Эти пары будут применены к файлу в этом входном каталоге. Если во входном каталоге обнаружен xml-файл Request, он открывается и считывается.

Последовательно просматриваются все элементы <Format> внутри элемента <ProcessingFormats> (ProcessingFormats/Format). Если во входном xml – файле присутствуют пространства имен (атрибут xmlns), они описываются в элементах <Namespace> (Format/Namespaces/Namespace). Для каждого формата анализируются элементы внутри элемента <FindData>, (например, как показано выше, <ComplexData> с ключом <Key>PLSC</Key>). Текстовое значение этого элемента формируется как объединение значений полей, путь до которых во входном Request-файле задается в элементе <XPath>. Атрибутами указывается тип данных во входном файле: текст (атрибут Type="Text") или атрибут (атрибут Type="Attribute", атрибут Name="имя атрибута"). Если элементы, описанные в <FindData>, найдены во входном файле и имеют не пустые значения, считается, что формат входного файла

определен, просмотр элементов <Format> заканчивается, начинается обработка входного файла по этому формату. Считанные из входного файла данные добавляются к списку ключ/значение, который сформировался при анализе элемента <Dir>. Ключем является имя элемента поиска <Key>PLSC</Key>, значением – текстовое значение элемента <ComplexData>. Внутри <FindData> должен быть хотя бы один элемент, но, если у него определен атрибут Required="**false**" и данные по этому пути не найдены, этот входной файл будет обрабатываться этим форматом, а значения для поиска во внешнем файле ExternalData.xml могут браться только из привязанных к каталогу.

Определяется файл настроек, который будет управлять процессом подготовки данных. Путь к нему описывается в элементе <ExternalFile> (ProcessingFormats/Format/ExternalFile). Считывается файл внешних настроек ExternalData.xml – файл, определенный в элементе <File> . Название и формат файла внешних настроек может быть любым, но должен соответствовать структуре пути для поиска в нем. Один из примеров поиска во внешнем файле – определение файла настроек. Путь к имени файла настроек в файле внешних настроек определяется в элементе <SettingsPath> (PaymentSystem/PLSC/Settings). Это означает, что в файле внешних настроек внутри корневого элемента должен быть элемент <PaymentSystem> с атрибутом Value, внутри него должен быть элемент <PLSC> с атрибутом Value, внутри него должен быть элемент <Settings>. У последнего элемента в пути (в данном случае 'Settings') должно быть текстовое значение. В данном случае значением элемента является имя файла настроек. Для поиска в значение атрибута Value подставляется соответствующее значение из списка ключ/значение, сформированного ранее. В данном случае значение атрибута 'Value' для элемента 'PaymentSystem ' берется из ключа 'PaymentSystem ', сформированном при определении каталога, а значение атрибута 'Value' для элемента 'PLSC ' берется из ключа 'PLSC', сформированном при определении формата файла. Такой поиск и такие пути применяется ко всем данным, которые можно взять из файла внешних настроек.

В каталоге, заданном в элементе <PathSettings>, по определенному имени ищется файл настроек, загружается, в соответствии с ним происходит процесс подготовки данных (описание файла настроек см. в след. разделе). В процессе подготовки данных формируется входной файл для программы CDP на основе данных о картах в Request-файле. Для каждой карты создается отдельная строка, в которую помещаются поля данных, разделенных символом '#'. Большая часть полей берется в неизменном виде из соответствующих элементов входного Request-файла (например, PAN карты). К некоторым полям применяется форматирование (например, вырезание подстроки). Есть поля, значения которых вычисляются на основе информации из других полей. К ним относятся поля, значения которых берутся из ответа на запрос в ЦИТ (номер СНИЛС, номер ОМС, Клиент ID, код льготника). Поле ПИН-блока формируется на основе PAN карты в результате соединения с HSM Thales. На основе информации о серии и номере паспорта клиента определяется полный путь до файла с фотографией и заносится в соответствующее поле. Параллельно с созданием входного файла для CDP формируется ответный Response-файл в формате XML. Большая часть информации для него берется из входного Request-файла. Другие данные определяются в процессе подготовки (например, ПИН-блок).

Сформированный файл для CDP выкладывается во входной каталог, определенный в элементе <PathOut>. Далее происходит запуск консольной программы CDP, в которую загружается шаблон настроек карточного продукта (ini-файл), определенный в параметрах Пластик Кода PLSC файла ExternalData.xml (см. описание в след. разделе). Программа CDP открывает сформированный входной файл с данными по картам и выполняет подготовку

чиповых данных. Подготовка данных в зависимости от количества карт во входном файле может занять значительное время. В результате подготовки данных CDP формирует два файла: файл для эмбоссирования и файл для печати ПИН-конвертов. Эти файлы выкладываются в определенные каталоги, откуда их дальше подхватывает DPService и формирует на их основе Common-файлы для эмбоссирования и для печати ПИН-конвертов. Common-файлы помещаются в каталоги, определенные в ExternalData.xml. Названия Common-файлов являются настраиваемыми. Правила их формирования описаны в файле настроек. Таким образом можно формировать Common-файлы, которые будут накапливать в себе данные по картам не из одного Request-файла, а сразу из нескольких. В процессе переноса данных по картам в Common-файл DPService выполняет обновление статуса карт в ЦИТ. В конце производится сортировка карт по правилу, описанному в файле ExternalData.xml.

4.2. Внешние настройки (ExternalData.xml)

Файл внешних настроек определяется в элементе ProcessingFormats/Format/ExternalFile/File. Название и формат файла внешних настроек может быть любым, но должен соответствовать структуре пути для поиска в нем (см. выше).

Пример файла внешних настроек:

```
<PaymentSystem Value="In">
  <PLSC Value="520935KGF" Description="">
    <Settings>OW_AKB_KGF.xml</Settings>
    <Ini>C:\OstPack\CDP\Ini\MC_PayPass+Social_KGF.ini</Ini>
    <FileOut>DPS_KGF.txt</FileOut>
    <FileForCDP>DPS_CDP_KGF.txt</FileForCDP>
    <PathOut>C:\OstPack\CDP\Input</PathOut>
    <FileOutAction>Move</FileOutAction>
    <EncodingOut>1251</EncodingOut>
    <PathResult>C:\OstPack\CDP</PathResult>
    <FileResult>err.txt</FileResult>
    <PathData>C:\OstPack\CDP\Output\Emboss</PathData>
    <FileNameData>CDP_EMB_KGF.txt</FileNameData>
    <FileDataAction>Move</FileDataAction>
    <PathDataCommon>C:\OstPack\DPService\Common\Emboss</PathDataCommon>
    <EncodingData>1251</EncodingData>
    <EncodingDataCommon>1251</EncodingDataCommon>
    <PathPin>C:\OstPack\CDP\Output\Pin</PathPin>
    <FileNamePin>CDP_PIN_KGF.txt</FileNamePin>
    <FilePinAction>Move</FilePinAction>
    <PathPinCommon>C:\OstPack\DPService\Common\Pin</PathPinCommon>
    <EncodingPin>1251</EncodingPin>
    <EncodingPinCommon>1251</EncodingPinCommon>
    <PhotoPathIn>\\10.128.19.6\Inhabitant\photo</PhotoPathIn>
    <PhotoPathOut>\\10.128.19.6\Inhabitant\photo</PhotoPathOut>
    <PhotoFileAction>None</PhotoFileAction>
    <ChangeStatus>IN_PROGRESS,CREATED</ChangeStatus>
    <InitialStatus>DENIED</InitialStatus>
    <!-- Data,Pin,Proc -->
    <StatusData>Data</StatusData>
    <IndexInfo>20</IndexInfo>
    <IndexSNILS>17</IndexSNILS>
     <SortData>true</SortData>
    <!-- Data,Pin,Proc -->
    <SortFileSource>Data</SortFileSource>
    <SortFieldsIndex>16,3,10,11,12</SortFieldsIndex>
  </PLSC>
</PaymentSystem>
```

Элемент <PaymentSystem> с атрибутом Value="In" здесь соответствует элементу <PaymentSystem>In</PaymentSystem>, привязанному к описанию входного каталога WorkingDirectories/Dir в файле DPService.xml.
Элемент <PLSC> с атрибутом Value="520935KGF" соответствует описанию элемента ProcessingFormats/Format/FindData/ComplexData с ключом <Key>PLSC</Key> в файле DPService.xml, где первые 6 цифр PAN объединяются со значением поля **ADD_INFO_1**. В атрибуте Description="" можно указать описание данного карточного продукта.
В элементе <Settings> определяется название файла настроек для обработки данного карточного продукта. Путь до этого элемента PaymentSystem/PLSC/Settings соответствует значению элемента ProcessingFormats/Format/ExternalFile/SettingsPath в файле DPService.xml.
<Ini> – название ini-файла с шаблоном настроек карточного продукта, который загружается в консольную программу CDP в процессе подготовки чиповых данных
<FileOut> – название файла с данными по картам для программы CDP, который формирует DPService
<FileForCDP> – название входного файла CDP, в который копируется содержимое файла <FileOut> перед запуском консольной CDP
<PathOut> – каталог, куда помещается файл <FileForCDP>
<FileOutAction> – действие, которое нужно выполнить над файлом <FileOut> в результате подготовки чиповых данных программой CDP. Возможные значения:

None - оставить
Move - переместить в каталог 'Save' при успехе, в каталог 'Error' при ошибке
MoveError - переместить в каталог 'Error' при ошибке, при успехе оставить
MoveOK - переместить в каталог 'Save' при успехе, при ошибке оставить
Delete - удалить в любом случае
DeleteOKMoveError - удалить при успехе, при ошибке переместить в каталог 'Error'
DeleteOKNoneError - удалить при успехе, при ошибке оставить
DeleteErrorMoveOK - удалить при ошибке, при успехе переместить в каталог 'Save'
DeleteErrorNoneOK - удалить при ошибке, при успехе оставить

<EncodingOut> – кодировка, в которой должен быть сформирован файл <FileOut>
<PathResult> – каталог, где формируется файл с результатом работы программы CDP
<FileResult> – название файла с результатом работы программы CDP (успех или неуспех с сообщением об ошибке)
<PathData> – каталог, в котором CDP формирует файл для эмбоссирования с подготовленными чиповыми данными
<FileNameData> – название файла для эмбоссирования в каталоге <PathData>
<FileDataAction> – действие, которое нужно выполнить над файлом <FileNameData> в результате его обработки DPService (формирования Common-файла для эмбоссирования). Возможные значения см. выше в описании элемента <FileOutAction>.
<PathDataCommon> – каталог, в котором должен быть сформирован Common-файл для эмбоссирования. Правило формирования названия этого файла см. ниже в описании файла настроек обработки.
<EncodingData> – кодировка, в которой CDP формирует файл <FileNameData>
<EncodingDataCommon> – кодировка, в которой DPService формирует Common-файл для эмбоссирования

<PathPin> – каталог, в котором CDP формирует файл для печати ПИН-конвертов
<FileNamePin> – название файла для печати ПИН-конвертов в каталоге <PathPin>
<FilePinAction> – действие, которое нужно выполнить над файлом <FileNamePin> в результате его обработки DPService (формирования Common-файла для печати ПИН-конвертов). Возможные значения см. выше в описании элемента <FileOutAction>.
<PathPinCommon> – каталог, в котором должен быть сформирован Common-файл для печати ПИН-конвертов. Правило формирования названия этого файла см. ниже в описании файла настроек обработки.
<EncodingPin> – кодировка, в которой CDP формирует файл <FileNamePin>
<EncodingPinCommon> – кодировка, в которой DPService формирует Common-файл для печати ПИН-конвертов
<PhotoPathIn> – каталог, в котором размещаются фотографии клиентов
<PhotoPathOut> – каталог, в который перекладываются фотографии обработанных клиентов (если не задан, имя файла фотографии будет сформировано с путем <PhotoPathIn>)
<PhotoFileAction> – действие, которое производится над найденным файлом фотографии клиента. Возможные значения:

None - ничего не делать
CopyOut - скопировать в выходной каталог
MoveOut - переместить в выходной каталог
CopyDirIn - скопировать во входном каталоге в подкаталог с именем входного файла
MoveDirIn - переместить во входном каталоге в подкаталог с именем входного файла
CopyDirOut - скопировать в выходной каталог в подкаталог с именем входного файла
MoveDirOut - переместить в выходной каталог в подкаталог с именем входного файла
CopyOutCopyDirIn - скопировать в выходной каталог и скопировать во входном каталоге в подкаталог с именем входного файла
CopyOutMoveDirIn - скопировать в выходной каталог и переместить во входном каталоге в подкаталог с именем входного файла
MoveOutCopyDirIn - переместить в выходной каталог и скопировать во входном каталоге в подкаталог с именем входного файла
MoveOutMoveDirIn - переместить в выходной каталог и переместить во входном каталоге в подкаталог с именем входного файла
CopyDirInOut - скопировать и в входном каталоге и в выходном каталоге в подкаталог с именем входного файла
MoveDirInOut - переместить и в входном каталоге и в выходном каталоге в подкаталог с именем входного файла
Binary - записать в выходной файл для CDP не имя файла с путем, а сам файл
Binary... (любое из предыдущих, например BinaryMoveDirOut) - записать в выходной файл для CDP сам файл, а затем с ним сделать действие

<ChangeStatus> – изменение статуса карты в ЦИТ в процессе переноса данных из файла, подготовленного CDP, в Common-файл. Статусы перечисляются через запятую (в примере для карты сначала выставляется статус IN_PROGRESS, потом CREATED)
<InitialStatus> – первоначальный статус карты, который выставляется перед первым запросом данных гражданина из ЦИТ
<StatusData> – из какого файла (подготовленного программой CDP) брать данные для формирования параметров запроса на изменение статуса карты. Возможные значения:

Data - файл для эмбоссирования
Pin - файл для печати ПИН-конвертов
Proc - файл для процессинга (в текущих настройках не используется)

<IndexInfo> – индекс поля в файле, подготовленном CDP, в котором содержатся данные из элемента ADTA входного Request-файла (эти данные необходимы в запросе на изменение статуса)

<IndexSNILS> – индекс поля в файле, подготовленном CDP, в котором содержится СНИЛС (эти данные необходимы в запросе на изменение статуса)

<SortData> – сортировать ли данные по картам в Common-файлах

<SortFileSource> – какой Common-файл брать за основу в процессе сортировки (записи в остальных файлах будут сортироваться параллельно). Возможные значения:

Data - файл для эмбоссирования
Pin - файл для печати ПИН-конвертов
Proc - файл для процессинга (в текущих настройках не используется)

<SortFieldsIndex> – индексы полей для сортировки в Common-файле. Начинаются с 0, перечисляются через запятую. Сначала сортировка ведется по первому индексу, потом среди равных по второму индексу и т.д.

4.3. Настройки подготовки данных

Файлы настроек находятся в каталоге, определенном в элементе <PathSettings> в файле DPService.xml. После идентификации карточного продукта по коду пластика PLSC происходит загрузка файла настроек, определенного в элементе PaymentSystem/PLSC/Settings файла ExternalData.xml. Обработка входного Request-файла происходит согласно правилам, описанным в этом файле. Файл настроек состоит из нескольких групп элементов.

Адреса сторонних серверов (может отсутствовать):
Сервер ЦИТ используется для получения информации о гражданах (номер СНИЛС, номер ОМС, Клиент ID, код льготника) и обновления статуса карты:

```
<Server>
  <Name>CITIZENINFO</Name>
  <Adress>https://scard.tatar.ru</Adress>
  <Port>443</Port>
  <ID>…</ID>
  <Key>…</Key>
  <Login>…</Login>
  <Password>…</Password>
</Server>
```

HSM Thales используется для генерации ПИН-блока. В элементе <PEK002> указывается криптограмма ключа PEK, на котором будет зашифрован ПИН-блок. В элементе <PIN> указывается константное значение PIN-кода для карточных продуктов, которые выпускаются с предустановленным ПИН:

```
<Server>
  <Name>THALES</Name>
  <Adress>10.128.33.80</Adress>
  <Port>1500</Port>
  <PEK002>…</PEK002>
  <PIN>…</PIN>
</Server>
```

Сервер для получения данных транспортного приложения:

```
<Server>
```

```
    <Name>MIFAREMAP</Name>
    <Adress>https://10.128.38.1...</Adress>
    <Port>1500</Port>
  </Server>
```

Настройки работы с файлами (обязательно):

```
  <FileSettings>
    <XMLTemplate>RSP2H_Template.xml</XMLTemplate>
    <Delimiter>#</Delimiter>
    <InputFileAction>None</InputFileAction>
  </FileSettings>
```

В элементе <XMLTemplate> указывается название шаблона Response-файла в формате XML. Каталог с файлами-шаблонами определен в элементе <PathXMLTemplates> в файле DPService.xml.
Элемент <Delimiter> определяет разделитель полей в выходном файле для CDP. Значение может быть пустым, тогда поля записываются по позициям и в описании поля должен присутствовать элемент <OutputFormat>.
<InputFileAction> – действия со входным Request-файлом после его обработки. Возможные значения см. выше в описании одноименного элемента в файле DPService.xml .

Настройки поиска во входном xml (обязательно) :

```
  <XPathSettings>
    <Namespace>
      <Uri>http://namespaces.globalplatform.org/systems-messaging/1.0.0</Uri>
      <Prefix>gp</Prefix>
    </Namespace>
    <XPathParent>PMJobs/gp:GPMessage/gp:GPBody/gp:ApplicationDataNotification</XPathParent>
    <XPathParentOut>PMJobs/gp:GPMessage/gp:GPBody/gp:ApplicationDataNotification
</XPathParentOut>
    <Section>
      <Name>CommonADTA</Name>
<XPath>gp:ApplicationCommonData/gp:ApplicationData/gp:ICCData/gp:DataSet[@Name=
'ADTA']/gp:Data[@DataElement='#']</XPath>
    </Section>
    <Section>
      <Name>CommonENCD</Name>
<XPath>gp:ApplicationCommonData/gp:ApplicationData/gp:ICCData/gp:DataSet[@Name=
'ENCD']/gp:Data[@DataElement='#']</XPath>
    </Section>
...
    <Section>
      <Name>CRN2_EMVT</Name>
<XPath>gp:ApplicationDataPerCRN[2]/gp:ApplicationData/gp:ICCData/gp:DataSet[@Name=
'EMVT']/gp:Data[@DataElement='#']</XPath>
    </Section>
    <DataValue>
      <!-- Какой тип содержит данные (Attribute, Text). -->
      <Type>Attribute</Type>
      <!-- Наименование атрибута, содержащего данные -->
      <Name>Value</Name>
    </DataValue>
    <DataValueOut>
      <!-- Какой тип содержит данные (Attribute, Text). -->
      <Type>Attribute</Type>
      <!-- Наименование атрибута, содержащего данные -->
      <Name>Value</Name>
    </DataValueOut>
  </XPathSettings>
```

Элементы <XPathParent> и <XPathParentOut> задают путь к элементу ApplicationDataNotification, который описывает одну карту во входном (Request) и выходном (Response) XML-файлах. Все дальнейшие пути будут строиться относительно этих.
В элементах <Section> определяются псевдонимы путей. Например вместо длинного пути в XML-файле
“gp:ApplicationCommonData/gp:ApplicationData/gp:ICCData/gp:DataSet[@Name='ADTA']/gp:Data[@DataElement='#']” в дальнейшем можно пользоваться его псевдонимом CommonADTA.
Элементы <DataValue> и <DataValueOut> определяют, что значения полей с данными карты по умолчанию нужно считывать из входном Request-файла и записывать в выходной Response-файл в атрибут ‘Value’.

Формирование имени выходного Response-файла (обязательно):
Имя выходного Response-файла формируется с помощью функции "FileNameRsp", реализованной программно. Ее логика заключается во взятии имени входного Request-файла и замене в нем подстроки “REQ” на “RSP”.

```
  <FileNameRsp>
    <Fields>
      <Field Type="Text" Action="FileNameRsp">
        <Name>NameFileRsp</Name>
        <XPath>gp:ApplicationDataPerCRN/gp:CRN</XPath>
        <Required>false</Required>
      </Field>
    </Fields>
  </FileNameRsp>
```

Формирование имени выходного Common-файла для эмбоссирования (может отсутствовать):
Имя выходного Common-файла для эмбоссирования настраивается в широких пределах. В его состав можно включать константы, значения полей из входного Request-файла, дату-время обработки:

```
  <FileNameDataCommon>
    <Fields>
      <Field>
        <!-- Константа “C” -->
        <Name>EMB_File</Name>
        <Value>C</Value>
      </Field>
        <!-- Значение поля ADD_INFO_1 из входного Request-файла -->
         <Field Type="Attribute" Name="Value">
        <Name>ADD_INFO_1</Name>
         <XPath>gp:ApplicationDataPerCRN/gp:ApplicationData/gp:ICCData/gp:DataSet[@Name=
'ADTA']/gp:Data[@DataElement='ADD_INFO_1']</XPath>
      </Field>
       <Field>
        <!-- Константа “.” -->
        <Name>Point</Name>
        <Value>.</Value>
      </Field>
      <Field ListGetKey="DPA_RUNDATETIME" TypeData="DateTime" FormatOut="ddMMHHmm">
        <!-- Дата-время обработки в формате ddMMHHmm -->
        <Name>DPA_RUNDATETIME</Name>
        <Value></Value>
        <Required>true</Required>
      </Field>
      <Field>
        <!-- Расширение файла - константа “.000” -->
        <Name>Ext</Name>
        <Value>.000</Value>
      </Field>
    </Fields>
  </FileNameDataCommon>
```

Формирование имени выходного Common-файла для печати ПИН-конвертов (может отсутствовать):
Имя выходного Common-файла для печати ПИН-конвертов настраивается аналогично Common-файлу для эмбоссирования:

```
<FileNamePinCommon>
  <Fields>
    <Field>
      <!-- Константа “P” -->
      <Name>PIN_File</Name>
      <Value>P</Value>
    </Field>
     ...
  </Fields>
</FileNamePinCommon>
```

Данные выходного файла для CDP (обязательно) :
Поля, которые выгружаются в выходной файл для CDP, размещаются в элементе <Fields>.
Выгрузка значения одного поля осуществляется в элементе <Field>:

```
<Field>
  <!-- Срок действия карты -->
  <Name>EXDT</Name>
  <Path>CommonENCD/EXDT</Path>
  <Required>true</Required>
</Field>
```

Здесь в элементе <Name> указывается название поля. С таким названием это поле будет фигурировать в логе в случае ошибки. Поэтому рекомендуется давать полям уникальные названия.
В элементе <Path> указывается путь до элемента во входном Request-файле. В этом примере видно, что был применен псевдоним пути CommonENCD, благодаря которому путь выглядит короче и нагляднее. Само поле располагается в элементе EXDT, а его значение берется из атрибута Value.
Элемент <Required> показывает обязательность поля. Значение true говорит об обязательности поля. Если такого поля не окажется в Request-файле, возникнет ошибка и обработка прервется. Значение false говорит о необязательности поля. Если такого поля не окажется в Request-файле, оно будет выгружено в виде пустой строки.
К выгружаемым полям можно применять форматирование и они могут быть составными:

```
<Field>
  <!-- Название компании -->
  <Name>CMPN</Name>
  <Field>
    <Name>CMPN</Name>
    <Path>CommonEMBD/CMPN</Path>
    <SubStr Index="-5" Length="-32000"></SubStr>
    <OutputFormat Length="13" PadRight=" "></OutputFormat>
    <Required>false</Required>
  </Field>
  <Field>
    <Name>Deliver</Name>
    <Path>CRN1_ADTA/DELIVER</Path>
    <Required>true</Required>
  </Field>
</Field>
```

Здесь видно, что составное поле ‘Название компании’ состоит из двух полей CMPN и Deliver. К полю CMPN применено форматирование – отбрасывание последних 5 символов и выгрузка значения фиксированной длины 13 символов, дополненного справа пробелами.
Значения выгружаемых полей можно включать в детальный лог (настраивается элементом <DetailLog> в файле DPService.xml). Для этого к полю надо добавить атрибут LogName со значением, которое будет выводиться в лог в качестве названия поля:

```
<Field LogName="ПАН">
  <!-- PAN карты -->
  <Name>PAND</Name>
  <Path>CommonENCD/PAND</Path>
  <Required>true</Required>
</Field>
```

Есть поля, которые вычисляются динамически. Ниже приведен пример поля, значение которого извлекается из ответа на запрос в ЦИТ. Это получение информации о гражданине: номер СНИЛС, номер ОМС, Клиент ID, код льготника.

```
<Field ErrorType="NonCritical">
  <Name>SNILS</Name>
  <Field Type="Text" ErrorType="NonCritical" Save="true">
    <Name>SNILSSave</Name>
    <XPath>gp:DataProcessingServiceData/gp:SNILS</XPath>
    <Required>true</Required>
    <AField Action="GetSNILS" ErrorType="NonCritical">
      <Name>SNILSGet</Name>
      <Field>
        <Name>SNILSData</Name>
            <Field>
                 <!-- Фамилия клиента -->
                 <Name>F</Name>
                 <Path>CRN1_ADTA/F</Path>
                 <Required>true</Required>
            </Field>
            <Field>
                 <!-- Разделитель - константа “|” -->
                 <Name>DelimiterI</Name>
                 <Value>|</Value>
            </Field>
            <Field>
                 <!-- Имя клиента -->
                 <Name>I</Name>
                 <Path>CRN1_ADTA/I</Path>
                 <Required>true</Required>
            </Field>
            <Field>
                 <!-- Разделитель - константа “|” -->
                 <Name>DelimiterO</Name>
                 <Value>|</Value>
            </Field>
            <Field>
                 <!-- Отчество клиента -->
                 <Name>O</Name>
                 <Path>CRN1_ADTA/O</Path>
                 <Required>true</Required>
            </Field>
            <Field>
                 <!-- Разделитель - константа “|” -->
                 <Name>DelimiterD</Name>
                 <Value>|</Value>
            </Field>
            <Field>
                 <!-- День рождения клиента -->
                 <Name>D</Name>
                 <Path>CRN1_ADTA/D</Path>
                 <Required>true</Required>
```

```
                </Field>
                <Field>
                     <!-- Разделитель - константа “|” -->
                     <Name>DelimiterS</Name>
                     <Value>|</Value>
                </Field>
                <Field>
                     <!-- Пол клиента -->
                     <Name>Sex</Name>
                     <Path>CRN1_ADTA/GEND</Path>
                     <Required>true</Required>
                </Field>
                <Field>
                     <!-- Разделитель - константа “|” -->
                     <Name>DelimiterPS</Name>
                     <Value>|</Value>
                </Field>
                <Field>
                     <!-- Серия паспорта клиента -->
                     <Name>PS</Name>
                     <Path>CRN1_ADTA/P_S</Path>
                     <Required>true</Required>
                </Field>
                <Field>
                     <!-- Разделитель - константа “|” -->
                     <Name>DelimiterPN</Name>
                     <Value>|</Value>
                </Field>
                <Field>
                     <!-- Номер паспорта клиента -->
                     <Name>PN</Name>
                     <Path>CRN1_ADTA/P_N</Path>
                     <Required>true</Required>
                </Field>
                <Field>
                     <!-- Разделитель - константа “|” -->
                     <Name>DelimiterPD</Name>
                     <Value>|</Value>
                </Field>
                <Field>
                     <!-- Дата выдачи паспорта клиента -->
                     <Name>PD</Name>
                     <Path>CRN1_ADTA/P_DT</Path>
                     <Required>true</Required>
                </Field>
           </Field>
      </AField>
   </Field>
</Field>
```

Здесь включенный в поля атрибут ErrorType="NonCritical" показывает, что в случае ошибки при получении информации из ЦИТ, такая ошибка не будет считаться критической. Обработка на этой карте не прервется, просто эта карта будет исключена из дальнейшей обработки, хотя в Response-файл такая карта попадет. Вся информация об этой карте из входного Request-файла сохранится во временный XML-файл в каталоге XmlError, чтобы потом была возможность дообработать проблемные карты.

Атрибут Save="true" у поля SNILSSave показывает, что значение этого поля нужно сохранить во временный XML-файл в каталоге XmlError. Это делается для того, чтобы запрос информации о гражданине из ЦИТ всегда выполнялся только один раз, независимо от того, были ли ошибки при обработке карты DPService’ом. В случае успешного получения информации из ЦИТ, эти данные (номер СНИЛС, номер ОМС, Клиент ID, код льготника) сохраняются во временный XML-файл по пути, определенном в элементе <XPath>. Если в процессе дальнейшей обработки карты возникнет ошибка и эта карта будет исключена из обработки, то будет возможность повторно запустить обработку этой карты из временного XML-файла. В этом

случае информация о гражданине не будет запрашиваться повторно из ЦИТ. Она возьмется из XML-файла по пути <XPath>.

Такая логика реализуется благодаря использованию т.н. альтернативного поля в элементе <AField>. Значение поля AField возьмется в том случае, если не будут найдены данные по пути <XPath>. Как раз так происходит при первой попытке обработки карты. Сначала ищется информация во временном XML-файле по пути <XPath>. Такой информации не обнаруживается, тогда вычисляется поле AField. Поле AField расчитывается благодаря вызову функции Action="GetSNILS", реализованной программно. Все оставшиеся поля, включенные в элемент <AField>, представляют из себя параметры запроса в ЦИТ.

Формирование данных транспортного приложения происходит в поле:

```
<Field Action="TransportApplication" ErrorType="NonCritical">
  <Name>TransportApplication</Name>
  <Field ErrorType="NonCritical">
    <Name>TransportApplicationData</Name>
    <Field ErrorType="NonCritical">
      <Name>MifareMap</Name>
      <Field Type="Text" ErrorType="NonCritical" Save="true">
        <Name>MifareMapSave</Name>
        <XPath>gp:DataProcessingServiceData/gp:MifareMap</XPath>
        <Required>true</Required>
        <AField Action="GetMifareMap" ErrorType="NonCritical" FromID="false">
          <Name>MifareMapGet</Name>
          <Field ErrorType="NonCritical">
            <Name>MifareMapData</Name>
            <Field>
                <!-- Срок действия карты -->
              <Name>EXDT</Name>
              <Path>CommonENCD/EXDT</Path>
              <Required>true</Required>
            </Field>
            <Field>
                <!-- Разделитель - константа “|” -->
              <Name>Delimiter1</Name>
              <Value>|</Value>
            </Field>
            <Field>
                <!-- Номер отделения доставки -->
              <Name>DELIVER</Name>
              <Path>CRN1_ADTA/DELIVER</Path>
              <Required>true</Required>
            </Field>
            <Field>
                <!-- Разделитель - константа “|” -->
              <Name>Delimiter2</Name>
              <Value>|</Value>
            </Field>
            <Field Type="Text" ErrorType="NonCritical">
                <!-- Код льготника -->
              <Name>CODEL</Name>
              <XPath>gp:DataProcessingServiceData/gp:SNILS</XPath>
              <Index>3</Index>
            </Field>
            <Field>
                <!-- Разделитель - константа “|” -->
              <Name>Delimiter3</Name>
              <Value>|</Value>
            </Field>
            <Field Type="Text" ErrorType="NonCritical">
                <!-- Клиент ID -->
              <Name>CLID</Name>
              <XPath>gp:DataProcessingServiceData/gp:SNILS</XPath>
              <Index>2</Index>
            </Field>
            <Field>
```

```
                <!-- Разделитель - константа “|” -->
              <Name>Delimiter4</Name>
              <Value>|</Value>
            </Field>
            <Field>
                <!-- Card ID -->
              <Name>CARD_ID</Name>
              <Path>CRN2_ADTA/CARD_ID</Path>
              <Required>true</Required>
            </Field>
          </Field>
        </AField>
      </Field>
    </Field>
    <Field>
      <!-- Разделитель - константа “|” -->
      <Name>Delimiter5</Name>
      <Value>|</Value>
    </Field>
    <Field>
      <!-- Скрипт для льготника -->
      <Name>SCRIPTL</Name>
      <Value>[SCRIPT=Gem_BENE.000]</Value>
    </Field>
    <Field>
      <!-- Разделитель - константа “|” -->
      <Name>Delimiter6</Name>
      <Value>|</Value>
    </Field>
    <Field>
      <!-- Скрипт для удобного -->
      <Name>SCRIPTN</Name>
      <Value>[SCRIPT=Gem_COMF.000]</Value>
    </Field>
  </Field>
</Field>
```

Значение поля формируется в результате вызова функции Action="TransportApplication". Само поле составное и формирует параметры этой функции. В его составе есть поле AField с вызовом функции Action="GetMifareMap", которая выполняет непосредственно сам запрос к серверу и возвращает данные транспортного приложения. Здесь реализован тот же механизм, как и при запросе данных из ЦИТ. Полученные в ответе от сервера данные транспортного приложения сохраняются во временный XML-файл по пути, определенном в элементе <XPath>. Атрибут FromID поля AField показывает, откуда брать код территории:
true - из ответа ЦИТ
false - из справочника (путь до которого определен в элементе <PathTerritoryData> в файле DPService.xml)

В составном поле MifareMapData определены параметры запроса. Поля CODEL (Код льготника) и CLID (Клиент ID) берутся из ранее сохраненных во временный XML-файл данных о гражданине, полученных из ЦИТ, по соответствующим индексам (значения номер СНИЛС, номер ОМС, Клиент ID, код льготника разделены символом # и нумеруются с 0). Поэтому запрос данных транспортного приложения должен идти после получения информации из ЦИТ.

В состав поля с Action="TransportApplication" входят названия скриптов персонализации в полях SCRIPTL (скрипт для льготного проездного) и SCRIPTN (скрипт для удобного проездного). Тот или иной скрипт выбирается в зависимости от значения поля CODEL (Код льготника). В конечном итоге функцией "TransportApplication" формируется поле со значением, содержащим название скрипта персонализации, поля для формирования отчета и данные транспортного приложения Mifare, разделенные символом табуляции.

ПИН-блок формируется в поле:

```
<Field>
  <Name>PIN</Name>
```

```
      <Field Type="Text" Save="true">
        <Name>PINSave</Name>
        <XPath>gp:DataProcessingServiceData/gp:PIN</XPath>
        <Required>true</Required>
        <AField Action="GetPIN" ErrorType="Critical">
          <Name>PINGen</Name>
          <Field>
            <!-- PAN карты -->
            <Name>PAND</Name>
            <Path>CommonENCD/PAND</Path>
            <Required>true</Required>
          </Field>
        </AField>
      </Field>
    </Field>
```

ПИН-блок генерируется полем AField с вызовом функции Action="GetPIN". В параметры функции передается PAN карты, который включается в запрос к HSM Thales на генерацию ПИН-блока. Эта функция анализирует значение элемента настроек <RandomPIN>. Если его значение true, генерируется случайный ПИН. Если его значение false, ПИН берется константой. Реализован механизм сохранения сгенерированного ПИН-блока во временный XML-файл по пути <XPath>gp:DataProcessingServiceData/gp:PIN</XPath>. При первой попытке обработать карту, ПИН-блок реально генерируется и сохраняется в XML-файл. Этот же ПИН-блок помещается в Response-файл. При повторных попытках обработать карту, берется ранее сохраненное значение ПИН-блока. Это сделано для того, чтобы ПИН-блок, выгруженный в Response-файл, не расходился с ПИН-блоком, участвующим в реальной подготовке чиповых данных и персонализации.

В отличие от запроса данных гражданина и транспортного приложения полю Afield здесь выставлен атрибут ErrorType="Critical". Это значит, что, в случае ошибки генерации ПИН-блока на HSM Thales, обработка на данной карте полностью прервется. В этом случае уже невозможно подготовить Response-файл по всем картам из входного Request, т.к. поле ПИН-блока в нем является обязательным. Это в отличие от запроса данных гражданина и транспортного приложения, т.к. эта информация не включается в Response-файл и независимо от наличия этих данных можно сформировать Response.

Блок данных в элементе ADTA из входного Request-файла выгружается полностью в формате XML в поле:

```
   <Field Type="Element" Encoding="Base64">
      <Name>ADTA_Xml</Name>
      <XPath>gp:ApplicationDataPerCRN[1]/gp:ApplicationData/gp:ICCData/gp:DataSet[@Name=
'ADTA']</XPath>
    </Field>
```

Атрибут Type="Element" указывает, что необходимо выгрузить полностью весь элемент со всем содержимым. Атрибут Encoding="Base64" определяет кодировку, в которой будет выгружено поле. В элементе <XPath> указан путь до элемента.

Проверка наличия файла фотографии клиента и выгрузка полного пути до него осуществляется в поле:

```
   <Field Action="GetPhoto" ErrorType="NonCritical">
      <Name>Photo</Name>
      <Field>
        <Name>PhotoData</Name>
        <Field>
          <!-- Серия паспорта клиента -->
          <Name>PS</Name>
          <Path>CRN1_ADTA/P_S</Path>
```

```
        <Required>true</Required>
      </Field>
      <Field>
        <!-- Любые символы по маске "*" -->
        <Name>DelimiterPN</Name>
        <Value>*</Value>
      </Field>
      <Field>
        <!-- Номер паспорта клиента -->
        <Name>PN</Name>
        <Path>CRN1_ADTA/P_N</Path>
        <Required>true</Required>
      </Field>
      <Field>
        <!-- Расширение файла любое по маске ".*" -->
        <Name>Ext</Name>
        <Value>.*</Value>
      </Field>
    </Field>
  </Field>
```

Поле вызывает функцию Action="GetPhoto", которая выполняет действие, определенное в настройке <PhotoFileAction> в файле ExternalData.xml. В данном случае это проверка наличия файла фотографии клиента и формирование полного пути до него. Тип ошибки указан "NonCritical". Это значит, что, в случае отсутствия фотографии, обработка не прервется полностью на этой карте, просто она будет исключена из дальнейшей обработки (при этом в Response эта карта попадет), а информация о ней сохранится во временный XML-файл. Когда появится файл фотографии, такую карту можно будет дообработать.

В параметр функции "GetPhoto" надо передать маску для поиска файла фотографии, которая формируется составным полем PhotoData.

Формирование Response-файла (обязательно) :

Формирование Response-файла начинается с создания его заголовка в элементе <Header>, который состоит из вложенного элемента <Fields>, в котором перечислены поля такого формата:

```
  <Field Type="Text" TypeOut="Text" Dynamic="false">
    <!-- Версия файла -->
    <Name>Version</Name>
    <XPath>FileHeader/Version</XPath>
    <XPathOut>FileHeader/Version</XPathOut>
  </Field>
```

В элементе <XPath> задается абсолютный путь, откуда брать значение поля из входного Request-файла. В элементе <XPathOut> задается абсолютный путь, куда записывать значение поля в выходной Response-файл.

Данные карт выгружаются в полях вида:

```
  <Field TypeOut="Attribute" NameOut="Value" OnlyXml="true">
    <!-- PAN карты -->
    <Name>PAND</Name>
    <Path>CommonENCD/PAND</Path>
    <PathOut>CommonADTA/PAND</PathOut>
    <Required>true</Required>
  </Field>
```

Можно понять, что значение поля выгружается в атрибут с именем ‘Value’. Атрибут OnlyXml="true" указывает, что поле нужно выгрузить только в выходной Response-файл формата XML. Если этого атрибута не будет или он будет равен false, то это поле выгрузится не только в Response-файл, но и во входной файл для CDP, что может оказаться ошибкой. В элементе <Path> задается относительный путь, откуда брать значение поля из входного Request-файла. В элементе

<PathOut> задается относительный путь, куда записывать значение поля в выходной Response-файл. Эти пути берутся относительно элемента <ApplicationDataNotification>, определяющего данные по одной карте. Элемент <Required> устанавливает обязательность поля.

ПИН-блок выгружается в поле:

```
<Field TypeOut="Attribute" NameOut="Value" OnlyXml="true">
  <Name>PIND1</Name>
  <Field Type="Text" OnlyXml="true">
    <Name>PIND_S</Name>
    <XPath>gp:DataProcessingServiceData/gp:PIN</XPath>
    <Required>true</Required>
  </Field>
  <PathOut Type="Attribute" Name="Value">CommonADTA/PIND</PathOut>
</Field>
```

Видно, что берется ранее сохраненное значение ПИН-блока во временном XML-файле по пути <XPath>gp:DataProcessingServiceData/gp:PIN</XPath>.

4.4. Логика обработки ошибок

Обработка входных Request-файлов начинается с их предварительной проверки тестовым модулем [Xml конвертер - Test]. Для каждой карты в Request-файле по сформированному коду пластика PLSC выполняется поиск соответствующих настроек карточного продукта в файле ExternalData.xml. Если настройки найти не удается (например, по причине отсутствия PLSC с таким значением), то такой входной файл сразу перекладывается в каталог Error и тем самым исключается из реальной подготовки данных. В лог записывается соответствующее сообщение об ошибке. В случае успешного определения файла настроек, он загружается, и согласно ему происходит тестовый прогон обработки. При этом выполняется только проверка наличия обязательных полей в Request-файле. Если какое-либо обязательное поле отсутствует, то об этом сообщается в логе и файл перекладывается в каталог Error.

Далее запускается боевой обработчик – модуль [Xml конвертер]. Request-файлы обрабатываются по одному, для каждой карты загружается соответствующий файл настроек. Определяется имя выходного Response-файла и в каталоге для ответных файлов выполняется проверка на наличие в нем файла с таким именем. Если такой ответный файл уже существует, считается, что была предпринята попытка обработать уже обработанный файл. В этом случае входной Request-файл перекладывается в каталог Error и в логе выводится соответствующее сообщение. В процессе боевой обработки могут возникнуть 4 вида ошибки:

1) Некритичная ошибка при получении информации о гражданине из ЦИТ, при запросе данных транспортного приложения или при проверке наличия файла фотографии. В этом случае обработка на этой карте не прервется, оставшиеся карты продолжат обрабатываться. Просто эта карта будет исключена из дальнейшей обработки (подготовки чиповых данных программой CDP, выгрузки в Common-файл и обновления статуса карты в ЦИТ). Вся информация об этой карте из Request-файла занесется во временный XML-файл в каталоге XmlError. Этот файл будет иметь название, совпадающее с названием входного Request-файла, только к нему еще добавится окончание “.err”. Однако такая карта попадет в Response-файл. Когда причины некритичной ошибки будут устранены, временный XML-файл можно дообработать. Для этого его нужно просто переложить во входной каталог и запустить обработку. Благодаря окончанию “.err” DPService поймет, что формировать Response-файл для такого входного файла уже не нужно. Вся ранее полученная информация из ЦИТ повторно запрашиваться не будет, потому что она была сохранена в этом временном XML-файле. Если при повторной обработке ошибок больше не возникает, то для такой карты выполняется подготовка чиповых данных программой CDP, выгрузка в Common-файл и обновление статуса в ЦИТ. Если ошибки повторяются

или возникают другие, то эта карта исключается из дальнейшей обработки, а информация о ней повторно сохраняется во временном XML-файле в каталоге XmlError. И так до тех пор, пока успешно не пройдет обработка всех карт.

2) Критичная ошибка при генерации ПИН-блока для карты. В этом случае обработка такого Request-файла полностью прерывается, оставшиеся карты игнорируются. Request-файл перекладывается в каталог Error. В лог записывается сообщение об ошибке. Response-файл и Common-файлы не формируются, подготовка чиповых данных программой CDP и обновление статуса карты в ЦИТ не выполняется. Надо устранить причину критичной ошибки и повторно выложить Request-файл во входной каталог.
3) Ошибка подготовки чиповых данных программой CDP. В этом случае обработка такого Request-файла полностью прерывается и он перекладывается в каталог Error. В лог записывается сообщение об ошибке. Response-файл и Common-файлы не формируются, обновление статуса карты в ЦИТ не выполняется. Надо устранить причину ошибки в программе CDP и повторно выложить Request-файл во входной каталог.
4) Ошибка при обновлении статуса карт в ЦИТ в процессе формирования Common-файлов. В этом случае карты, для которых не удалось обновить статус, заносятся в отдельный Common-файл в каталоге Error. В основной Common-файл такие карты не попадают. В основной Common-файл переносятся только полностью успешно обработанные карты, готовые к эмбоссированию и печати ПИН-конвертов. Причина ошибки обновления статуса выведется в лог. После ее устранения обновлять статус проблемных карт из Common-файлов в каталоге Error придется с помощью сторонних средств.