Authorization Gateway
Specification & Implementation Document

Version 2.8
06 20 2014

1.0 Introduction

This document serves as a technical guide for transmitting check transactions to Sage and details the communication method, authorization request specifications, and response specifications. Its purpose is to provide software developers with the necessary information to create an interface for check authorization.

Sample code is provided at the end of this document in VB.NET and C#. In addition, a complete Authorization Gateway sample solution is available to help further illustrate how to create an interface that uses the Authorization Gateway.

2.0 Overview

The Authorization Gateway is designed to accommodate various input requirements based on a given terminals settings. This allows for the development of a single interface that can be easily configured to handle many different scenarios.

The Authorization Gateway uses web services to present distributed methods for integration into client applications, and an interface with the Authorization Gateway can be developed with any programming language that can consume a web service.

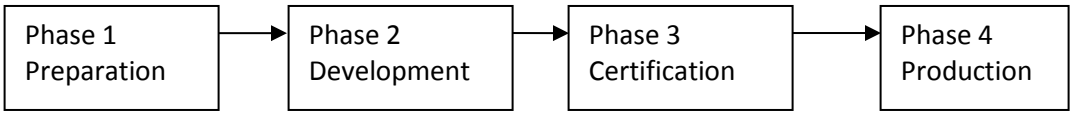
Extensible Markup Language (XML) is used to send data packet requests to the Authorization Gateway and to receive a response back. Simple Object Access Protocol (SOAP) is used for XML message exchange over HTTPS. The Authorization Gateway also employs a custom SOAP header for authentication information.

XML Schema Definitions (XSDs) are used by the Authorization Gateway to validate data packet requests sent by the client. Each terminal will be assigned a published XSD based on the terminal settings. If a data packet request does not conform to its assigned XSD a failed Validation Message response will be returned, otherwise the data packet will be processed as requested.

3.0 Workflow Overview

This document will provide a full lifecycle workflow for integrating the authorization gateway into your host system. This workflow was designed to make the gateway integration process as simple as possible and will be used to guide your team throughout the integration effort and to keep the project on track.

The workflow consists of four distinct phases, with each phase culminating in a major milestone. The four distinct phases of the workflow are Preparation, Development, Certification, and Production. The linear graphic below illustrates how each of these phases leads to the next.



These four distinct phases identify the critical planning, assessment, and coordination of activities between the integration team and Sage's software development team.

Each phase is marked with a single major milestone that represents the successful culmination of all the activities of the phase. In addition to this major event, each phase may also have intermediate milestones leading up to the major milestone. These events mark the self-regulation points of the process. They are review and synchronization points rather than freeze points. They represent points in time when all team members synchronize the integration effort, and members of the project team agree that they have achieved the objectives of that particular phase. Milestones allow the team to assess the status of the integration effort as well as to make any necessary adjustments in the project scope to accommodate any changes that have developed during the course of the integration effort.

The individual phases are outlined below and include a table that defines the milestones required to complete each phase. The tables can also be used by your team to chart the progress of the integration effort.

3.1 Phase 1: Preparation

The Preparation Phase is the initial phase of the integration effort. During this phase the integration team will be responsible for reviewing and understanding this document as well as completing the milestones listed below. The completion of this phase marks the opportunity for all to evaluate the integration effort, identify any remaining issues, and begin the Development Phase.

Phase	Milestone	Completed
Preparation	Review the Authorization Gateway Specification document	
	Obtain a User Name and Password for Certification	
	Determine your SEC Code(s)	
	Determine your XML Template(s)	
	Determine your XML Schema(s)	
	Determine your Certification Terminal ID(s)	
	Establish Connectivity	
	Request Certification Terminal Settings	

3.1.1 Preparation Phase Milestones:

- **Review the Authorization Gateway Specification:** This document details the communication method, authorization request specifications, and response specifications.
- **Obtain a User Name and Password for Certification:** In order to connect to the Authorization Gateway you must contact us to obtain a user name and password that will allow you to connect to the gateway to begin the integration effort. This user name and password will be unique to your team and will only allow you to invoke web methods used for certification. Once you have reached the Production Phase you will be given another user name and password that will allow you to invoke production web methods.
- **Determine your SEC Code(s):** The SEC Codes are defined later in this document and are the main factor in determining what XML Template and Schema to use.
- **Determine your XML Template(s):** Once you have determined your SEC Code you can determine which XML Template to use. The "How to determine which XML

Template to Use" section of this document explains the purpose of the XML templates and will assist you in determining which Template(s) to use.

- **Determine your XML Schema(s):** Once you have determined your SEC Code you can also determine which XSD will be used to validate your data packet submission. The "How to determine which XSD to Use" section of this document explains the purpose of the XSDs and will assist you in determining which XSD(s) to use.
- **Determine your Certification Terminal IDs:** Once you have determined your XSD(s), you can easily find the corresponding Certification Terminal ID listed in the same row as the XSD URL.
- **Establish Connectivity:** Create a web reference to the URL defined in the "Connection Method" section. This URL is only good for testing and certification. A production URL will be provided during the final phase of the integration effort.
- **Request Certification Terminal Settings:** Successfully invoke the GetCertificationTerminalSettings web method for each Certification Terminal ID previously identified.

3.2 Phase 2: Development

The Development Phase is the second phase of the integration effort. During this phase the integration team will be responsible for ensuring the host application can properly handle Authorizations, Declines, Voids, Reversals and in some cases Credits, Represented Checks and Manager Overrides. The section below entitled "Beginning Authorization Gateway Development" details the business logic necessary to complete each milestone in this phase. The completion of this phase marks the opportunity to begin the Certification Phase.

Phase	Milestone	Completed
Development	Validation Handling	
	Process Single Certification Check – Authorization	
	Process Single Certification Check – Check Limit Exceeded	
	Process Single Certification Check – Decline	
	Process Single Certification Check – Void a previously Authorized Check	
	Process Single Certification Check – Reversal	
	Process Single Certification Check – Credit (if applies)	
	Process Single Certification Check – Manager Needed (if applies)	
	Process Single Certification Check – Represented Check (if applies)	
	Exception Handling	

3.2.1 Development Phase Milestones:

- **Validation Handling:** Successfully validate a request Data Packet against your published XSD(s) and have the host system be able to handle failed validation messages.
- **Process Single Certification Check – Authorization:** Successfully invoke the ProcessSingleCertificationCheck web method and send a data packet with the necessary information to generate an Authorization response.

- **Process Single Certification Check – Check Limit Exceeded:** Successfully invoke the ProcessSingleCertificationCheck web method and send a data packet with the necessary information to generate a Check Limit Exceed response.
- **Process Single Certification Check – Decline:** Successfully invoke the ProcessSingleCertificationCheck web method and send a data packet with the necessary information to generate a Decline response.
- **Process Single Certification Check – Void:** Successfully invoke the ProcessSingleCertificationCheck web method and send a data packet with the necessary information to generate a Void response for a previously authorized check.
- **Process Single Certification Check – Reversal:** Successfully invoke the ProcessSingleCertificationCheck web method and send a data packet with the necessary information to generate a Reversal response.
- **Process Single Certification Check – Credit:** Successfully invoke the ProcessSingleCertificationCheck web method and send a data packet with the necessary information to generate an Authorization response.
- **Process Single Certification Check – Manager Needed:** Successfully invoke the ProcessSingleCertificationCheck web method and send a data packet with the necessary information to generate a Manager Needed response, and successfully perform an override.
- **Process Single Certification Check –Represented Check:** Successfully invoke the ProcessSingleCertificationCheck web method and send a data packet with the necessary information to generate a Represented Check Response, and successfully perform an override.
- **Exception Handling:** Include exception handling in the host system.
- **Request a Certification Script:** Upon successful completion of the above milestones you can request a certification script.

3.3 Phase 3: Certification

The Certification Phase is the third phase of the integration effort. During this phase the integration team will be responsible for sequentially completing the objectives outlined in a "Certification Script" that will be provided. Our integration team will closely monitor each transaction to ensure it is valid, and that the host system is properly configured to handle the various responses. The completion of this phase marks the opportunity to begin the Production Phase. However, if it is determined that the host system needs further refinements, it may be necessary to revert back to the Development Phase to make the necessary changes.

Phase	Milestone	Completed
Certification	Request Certification Script	
	Complete the Certification Script	

3.3.1 Certification Phase Milestones:

- **Request Certification Script:** Request a certification script from the Integration department.
- **Complete the Certification Script:** Successfully complete each objective defined in the certification script.

3.4 Phase 4: Production

The Production Phase is the final phase of the integration effort. During this phase the integration team will be responsible for configuring the host application for production. This includes obtaining the production URL and authorization credentials as well as completing the milestones listed below.

Phase	Milestone	Completed
Production	Request a User Name and Password for Production	
	Request the Production URL	
	Request a Production Terminal ID	
	Redirect the host application to use the Production Web Methods using the Production User Name, Password, and Terminal ID.	
	Request a "Go Live" Date	
	Request a User Name and Password for Production	

3.4.1 Production Phase Milestones:

- **Request a User Name and Password for Production**: Obtain a new unique user name and password that is authorized to invoke the production web methods.
- **Request the Production URL**: Obtain the URL that will be used to reference the production web methods.
- **Request a Production Terminal ID:** Obtain a Terminal ID for use in production.
- **Redirect the Host Application:** Redirect the host application to use the production URL and web methods with the provided production User Name, Password, and Terminal ID.
- **Request a "Go Live" Date:** Upon completion of the successfully submission of a single check a "Go Live" date for the host system can be established.

4.0 Preparing for Authorization Gateway Development (Phase 1)

4.1 Where Do I Start?

There are several milestones in the Preparation Phase that need to be completed prior to beginning development. These milestones were detailed in the section above, but your first step would be to obtain a user name and password.

It is assumed that you are familiar with working with XML, consuming web services, and with adding SOAP headers. Sample code is provided at the end of this document, and the "Submission" section of this document defines the SOAP header. If you have any questions or need any guidance please feel free to contact us at the number provided in the "Contact Information" section.

Once you have successfully connected to the Authorization Gateway and are comfortable with adding the SOAP header, the Preparation Phase culminates with the major milestone of invoking the GetCertificationTerminalSettings web method.

The GetCertificationTerminalSettings web method is defined in the “Terminal Settings – XML Specification” section, but essentially this web method can be invoked to request information about a specific certification terminal. This web method does not need to be invoked on a continuous basis, but can be invoked if your implementation team determines that the host system needs to acquire information about the Authorization Gateway Terminal. The invocation of this web method is made part of the Preparation Phase because it is the simplest web method and requires no input parameters.

It is important to note that the GetCertificationTerminalSettings has a sister web method called GetTerminalSettings that performs the same function for production terminals, but we will go into this in more detail during the Production Phase.

4.2 What are the different Standard Entry Class (SEC) Codes?

The Authorization Gateway uses the Standard Entry Class (SEC) codes to determine what information is required to be sent in the submission. The National Automated Clearing House Association (NACHA) requires the use of SEC Codes for each transaction settled through the Automated Clearing House (ACH). Each code identifies what type of transaction occurred. In addition, the SEC_CODE element in the response XML Data Packet form the GetCertificationTerminalSettings web method will include the SEC code used from the terminal ID provided. A definition of each of the SEC codes used by the Authorization Gateway can be found below.

- **Prearranged Payment and Deposit Entry (PPD):** A prearranged payment and deposit entry is either a standing or single entry authorization where the funds are transferred to or from a consumers account.
- **Corporate Credit or Debit (CCD):** A prearranged payment and deposit entry is either a standing or single entry authorization where the funds are transferred to or from a business account.
- **Internet Initiated Entry (WEB):** An internet initiated entry is a method of payment for goods or services made via the internet.
- **Telephone Initiated Entry (TEL):** A telephone initiated entry is a payment for goods or services made with a single entry debit with oral authorization obtained from the consumer via the telephone.
- **Point-of-Purchase Entry (POP):** The Point-of-Purchase method of payment is for purchases made for goods or services in person by the consumer. These are non-recurring debit entries. A check reading device must be used to capture the routing number, account number, and check number from the source document (check). The source document cannot be previously used for any prior POP entry, and the source document must be voided and returned to the customer at the point-of-purchase. In addition a signed receipt must be obtained at the point-of-purchase and retained for 2 years from the settlement date. The “Authorization Requirements” section in the Authorization Gateway Specification document contains additional information on the receipt requirements.
- **Check 21 (C21):** Although not an SEC Code C21 is used to denote Check 21 transactions. Check 21 requires a check reading device capture the routing number, account number, and check number from the source document (Check) as well as capture images of both the front and back of the source document.

5.0 Beginning Authorization Gateway Development (Phase 2)

5.1 I'm ready to begin Development. Where do I start?

The best place to start is with determining your application architecture for interfacing with the Authorization Gateway. At this point you have already determined which published XSD(s) your XML data packets will be validated against, and you also know the URL for the corresponding XML template(s) for your schema(s). This leaves you with the following possibilities for creating your XML data packets that are sent to the Authorization Gateway:

1. You can use an XML Schema Definition Tool (such as Xsd.exe for .Net or Svcutil.exe) to generate a class based on the published XSD, populate the class properties, and then serialize the object.
2. You can use LINQ to XML to build your xml and populate the elements and attributes.
3. You can load the XML template into an XML document object and use Xpath to populate the elements and attributes.
4. You can build your own XML document and use Xpath to populate the elements and attributes.

We recommend that you leverage the published XSDs and XML templates and use either the first or second options when creating the data packets to be sent. All of these methods are using the .NET platform however we have had several developers program to us using other languages.

We have provided example request XML Data Packets to assist your integration team with getting started. A link to these examples can be found at the end of the "How to determine which XML Template to Use" section.

5.2 What does it all mean?

Now that you have determined the best way to create the XML data packets within your host system, we will begin to look at what each element and attribute means, and when to apply a given value. The "Data Packet – XML Specification" section provides a detailed description of each element and includes a text description of the regular expressions, data types, or enumerations that control the allowed data formats for each element as well as links to the published data type XML templates.

5.3 How do I identify my data?

The specification for the Authorization Gateway XML Data Packet allows you to optionally identify your data in two distinct ways. Again, these settings are completely optional, but have been built in so that your host system can match a response from the Authorization Gateway with the original request. Since these identifiers are generated by the host system, the Authorization Gateway does not ensure that any optional identifiers contained in the XML Data Packet are unique. Rather the Authorization Gateway leaves the responsibility of determining if an identifier is unique to the host system. It is not required that optional identifiers are unique, but it is recommended. If an identifier is not unique it may become difficult for your host system to match responses or retrieve archived responses. In the examples provided, GUIDs have been used as optional identifiers. The use of GUIDs ensures uniqueness, but any value can be

used as an identifier, including database identity column values. It is also important to note that if the implementation team determines an identifier needs to be unique, that it only needs to be unique for a specific terminal ID, but it can be unique across all terminal IDs for a given user.

As previously mentioned there are two distinct ways you can optionally identify the XML Data Packet. These are the REQUEST_ID attribute contained within the AUTH_GATEWAY element and the TRANSACTION_ID element.

The **REQUEST_ID** attribute is a unique identifier that is used to identify the overall data packet. When your data packet is received by the Authorization Gateway it is not only processed, it is also asynchronously stored along with the response. This was done so that if necessary the host system can invoke the GetArchivedResponse web method to request a previous response. The GetArchivedResponse web method accepts the REQUEST_ID as an input parameter and will return the corresponding response. It is important to note that the GetArchivedResponse is a production only web method, and can only be effectively used if the host system keeps track of and submits values in the REQUEST_ID attribute. The value in the REQUEST_ID attribute of the request data packet is also returned in the response data packet in the REQUEST_ID attribute of the RESPONSE element.

The **TRANSACTION_ID** element is a unique identifier that is used to identify a specific transaction. The value contained in the TRANSACTION_ID element is recorded by the Authorization Gateway, but is not used internally and cannot be used to request a specific transaction. The value in the TRANSACTION_ID element is however returned in the response data packet in the TRANSACTION_ID element within the parent AUTHORIZATION_MESSAGE element. This was done so that your host system can match the response for a specific transaction to an internal record in the host system.

Again, setting values for both the REQUEST_ID attribute and the TRANSACTION_ID element are optional, but if the implementation team determines there is a business need to match records internal to the host system with Authorization Gateway responses or request archived responses; then these identifiers can be utilized to implement that functionality.

5.4 What do the different identifiers mean?

Each request XML Data Packet must contain a valid identifier for its schema. The identifier you use will change depending on the context of the transaction being sent. Your integration team will become more familiar with the different identifiers as you begin to work on each milestone. However, a list of all the valid identifiers can be found below.

- **Authorize (A**): This is used in schemas for POP, TEL, WEB, and Check 21 to indicate that an authorization is requested for the XML Data Packet being sent. It is also used to process credit transactions.
- **Recurring I:** This is used in schemas for PPD and CCD to indicate that an authorization is requested for a single or reoccurring transaction.
- **Void (V):** This is used in schemas for PPD, CCD, POP, TEL, WEB, and Check 21 to void a previously authorized transaction. However, it should be noted that transactions can only be voided on the same calendar day they were authorized.
- **Override (O):** This is used in schemas for POP, TEL, and Check 21 when the host system receives a manager needed message to void the previous transaction and input a new transaction in its place.

- **Payroll (P):** This is used in schemas for POP and Check 21 for business and payroll checks. What this does is NOT link the driver's license to the routing/ account numbers since the person writing/cashing the check is usually not the business.

5.5 What does verification only mean?

If the gateway terminal is setup as verification only or the VERIFICATION_ONLY element is set to true, then the transaction will be processed as verification only. This means that an authorization will be run, but that the check will not undergo Electronic Check Conversion (ECC) and that the check will have to be taken to the bank for deposit. In addition, depending on the merchants program, the funds may or may not be guaranteed.

5.6 What do I need to provide in the account section?

All PPD, CCD, TEL and WEB schemas define that the ACCOUNT child elements must contain values. The child elements within the ACCOUNT element for POP and Check 21 (C21) schemas define what ACCOUNT child elements must contain values and what ACCOUNT child elements can be left empty. All of the child elements within the ACCOUNT element for POP and Check 21 (C21), except the ACCOUNT_TYPE for POP schemas, define the data as optional. This is because for these SEC codes you can either provide the swiped MICR data or provide the routing, account, and check numbers. If the MICR_DATA, ROUTING_NUMBER, ACCOUNT_NUMBER, and CHECK_NUMBER are all left empty in the request data packet then the transaction cannot be processed. Either the MICR_DATA or the ROUTING_NUMBER, ACCOUNT_NUMBER, and CHECK_NUMBER elements must contain values.

It is important to note that if the swiped MICR data in the MICR_DATA element is missing, but the ROUTING_NUMBER, ACCOUNT_NUMBER, and CHECK_NUMBER elements contain values then the transaction will be processed as verification only; even if the CONTROL_CHAR indicates that the information was retrieved from a check reader. In addition, if the MICR_DATA, ROUTING_NUMBER, ACCOUNT_NUMBER, and CHECK_NUMBER elements all contain values, then the Authorization Gateway will only use the information in the MICR_DATA element and will parse it out overwriting any values sent in the ROUTING_NUMBER, ACCOUNT_NUMBER, and CHECK_NUMBER elements.

5.7 When do I need to include identity information?

Identity information needs to be included when the terminal is setup to do identity verification. There are schemas that will handle the validation for terminals that are setup to do identity verification, and the GetCertificationTerminalSettings web method will return a response of "true" in the RUN_IDENTITY_VERIFICATION element. If a terminal is setup to do identity verification then the host system is required to send either the last 4 of the check writers social security number OR their birth year (not both).

6.0 Connection Method

Sage supports connection via secure (https) webservice using SOAP. SOAP is a simple XML-based protocol to let applications exchange information over HTTP.
The webservice address used for certification and testing is as follows:

https://demo.eftchecks.com/webservices/AuthGateway.asmx

A username and password for certification will be provided.

NOTE: *A production webservice address, user name, and password will be supplied upon successful certification.*

7.0 Submission

The Authorization Gateway has been designed for fast and easy integration with your existing system. Simply request the Terminal Settings, complete the returned xml data packet template, and return it to the Authorization Gateway for processing. To accomplish this Authorization Gateway provides web methods for certification and for production. In addition, each web method contains a custom SOAP header used for authentication.

7.1 SOAP Header

The SOAP header contains the following fields:

UserName	String	Username provided by Sage for authorization.
Password	String	Password provided by Sage for authorization.
Terminal ID	Integer	Unique to each terminal used. Provided by Sage at time of terminal approval. Terminal IDs for certification are provided in this document.

8.0 Web Methods

A definition of the web methods can be found below. Each web method contains a hyperlink to a sample SOAP request and response.

8.1 Certification Web Methods

A definition of the certification web methods can be found below. Each web method contains a hyperlink to a sample SOAP request and response.

8.1.1 GetCertificationTerminalSettings

Description: This method will return the Terminal Settings for a certification Terminal. This method is used during interface testing and certification.
Input: Accepts no parameters.
Output: Outputs an XML string.

8.1.2 AuthGatewayCertification

Description: This method will validate that the interface is sending a data packet that conforms to its schema and is used during interface testing and certification.
Input: Accepts an XML string called a data packet that must conform to the terminals schema provided in the certification Terminal Settings.

Output: Outputs an XML string.

8.1.3 ProcessSingleCertificationCheck

Description: This method will run the authorization for a single certification check based on the settings for the provided certification terminal. A list of the valid certification routing numbers and their purpose is below. This method is used during interface testing and certification.

Routing Number	Purpose
490000018	Authorization
490000034	Decline
490000021	Manager Needed
490000047	Re-Presented Check

Input: Accepts an XML string called a data packet that must conform to the certification terminals schema provided in the certification Terminal Settings.
Output: Outputs an XML string.

8.2 Certification Web Methods when using Tokens

A definition of the certification web methods when using tokens can be found below. Each web method contains a hyperlink to a sample SOAP request and response.

8.2.1 GetCertificationTerminalSettings

Description: This method will return the Terminal Settings for a certification Terminal. This method is used during interface testing and certification.
Input: Accepts no parameters.
Output: Outputs an XML string.

8.2.2 AuthGatewayCertification

Description: This method will validate that the interface is sending a data packet that conforms to its schema and is used during interface testing and certification.
Input: Accepts an XML string called a data packet that must conform to the terminals schema provided in the certification Terminal Settings.
Output: Outputs an XML string.

8.2.3 ProcessSingleCertificationCheckWithToken

Description: This method will run the authorization for a single certification check based on the settings for the provided certification terminal using either, a given Token or the Account Type, Routing Number, and Account Number. A list of the valid certification routing numbers and their purpose is below. This method is used during interface testing and certification.

Routing Number	Token	Purpose
490000018	05944FB3E1DA4663868455AF630F45BE	Authorization
490000034	15944FB3E1DA4663868455AF630F45BE	Decline
490000021	25944FB3E1DA4663868455AF630F45BE	Manager Needed
490000047	35944FB3E1DA4663868455AF630F45BE	Re-Presented Check

Input: Accepts an XML string called a data packet that must conform to the certification terminals schema provided in the certification Terminal Settings.
Output: Outputs an XML string.

NOTE: *Using this method by passing the Account Type, Routing Number, and Account Number will create a TOKEN and pass it back in the Authorization Message Response. If a TOKEN already exists for the Account Type, Routing Number, and Account Number, the current TOKEN will be passed back in the Authorization Message Response.*

8.2.4 GetCertificationToken

Description: This method will return a Token for the Account Type, Routing Number, and Account Number.
Input: Accepts an XML string called a data packet that must conform to the certification terminals schema provided in the certification Terminal Settings
Output: Outputs an XML string.

8.2.5 ParseCertificationMICR

Description: This method will return an Account Type, Routing Number and Account Number.
Input: Accepts an XML string called a data packet that must conform to the certification terminals schema provided in the certification Terminal Settings
Output: Outputs an XML string.

8.3 Production Web Methods

A definition of the production web methods can be found below. Each web method contains a hyperlink to a sample SOAP request and response.

8.3.1 GetTerminalSettings

Description: This method will return the Terminal Settings for a terminal.
Input: Accepts no parameters.
Output: Outputs an XML string.

8.3.2 ProcessSingleCheck

Description: This method will run the authorization for a single check based on the settings for the terminal.

Input: Accepts an XML string called a data packet that must conform to the terminals schema provided in the Terminal Settings.
Output: Outputs an XML string.

8.3.3 GetArchivedResponse

Description: This method will retrieve a response for a previously processed transaction.
Input: Accepts a Request ID string.
Output: Outputs an XML string.

8.4 Production Web Methods when using Tokens

A definition of the production web methods when using tokens can be found below. Each web method contains a hyperlink to a sample SOAP request and response.

8.4.1 ProcessSingleCheckWithToken

Description: This method will run the authorization for a single check based on the settings for the terminal using either, a given Token or the Account Type, Routing Number, and Account Number.
Input: Accepts an XML string called a data packet that must conform to the terminals schema provided in the Terminal Settings.
Output: Outputs an XML string.

NOTE: *Using this method by passing the Account Type, Routing Number, and Account Number will create a TOKEN and pass it back in the Authorization Message Response. If a TOKEN already exists for the Account Type, Routing Number, and Account Number, the current TOKEN will be passed back in the Authorization Message Response.*

8.4.2 GetToken

Description: This method will return a Token for the Account Type, Routing Number, and Account Number.
Input: Accepts an XML string called a data packet that must conform to the terminals schema provided in the Terminal Settings.
Output: Outputs an XML string.

8.4.3 ParseMICR

Description: This method will return an Account Type, Routing Number and Account Number.
Input: Accepts an XML string called a data packet that must conform to the terminals schema provided in the Terminal Settings.
Output: Outputs an XML string.

9.0 Terminal Settings – XML Specification

The GetCertificationTerminalSettings and GetTerminalSettings web methods will return the following XML string.

9.1 Terminal Settings XML Example:

```
 <?xml version="1.0" encoding="utf-8"?>
<TERMINAL_SETTINGS xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:xsd="http://www.w3.org/2001/XMLSchema">
         <TERMINAL_ID>2318</TERMINAL_ID>
         <SEC_CODE>WEB</SEC_CODE>
         <IS_GATEWAY_TERMINAL>true</IS_GATEWAY_TERMINAL>
         <ALLOW_CNSMR_CREDITS>false</ALLOW_CNSMR_CREDITS>
         <DL_REQUIRED>false</DL_REQUIRED>
         <RUN_CHECK_VERIFICATION>false</RUN_CHECK_VERIFICATION>
         <RUN_IDENTITY_VERIFICATION>false</RUN_IDENTITY_VERIFICATION>
         <SCHEMA_FILE_PATH>
               http://localhost/geti.emagnus.webservices/Schemas/WEB/Ng_CheckNoVerificationDLOptionalWithIP.xsd
         </SCHEMA_FILE_PATH>
         <XML_TEMPLATE_PATH>
               http://localhost/geti.emagnus.webservices/Schemas/WEB/Templates/CheckNoVerificationDLOptionalWithIP.xml
         </XML_TEMPLATE_PATH>
</TERMINAL_SETTINGS>
```

The Terminal Settings XML will contain the following elements:

- TERMINAL_SETTINGS: Is the parent element and contains all other elements within the Terminal Settings XML document.
- TERMINAL_ID: Contains the ID for the terminal. The Terminal ID will be numeric value.
- SEC_CODE: Contains the Standard Entry Class. This will either be Ck21, PPD, CCD, POP, TEL, or WEB.
- IS_GATEWAY_TERMINAL: Contains true or false indicating if the Terminal is a gateway terminal or not.
- DL_REQUIRED: Contains true or false indicating if the terminal requires the driver's license state and number is to be included in the data packet request.
- RUN_CHECK_VERIFICATION: Contains true or false indicating if the terminal is setup for check verification.
- RUN_IDENTITY_VERIFICATION: Contains true or false indicating if the terminal is setup for identity verification.
- SCHEMA_FILE_PATH: Contains the Uniform Resource Identifier (URI) specifying the published XML Schema Definition (XSD) that the data packet request will be validated against.
- XML_TEMPLATE_PATH: Contains the Uniform Resource Identifier (URI) specifying the published XML template that can be used as the basis for creating the data packet request.

10.0 Validation Handling

When the AuthGatewayCertification web method receives a request it will first validate your request XML Data Packet against the published XSD for your terminal. Each returned response will include a VALIDATION_MESSAGE element. If the request XML Data Packet successfully passes validation the RESULT child element of the VALIDATION_MESSAGE element will

contain a value of “Passed”, but if the validation failed, the RESULT element will contain a value of “Failed”. These values can be coded into your host system for determining if a request passed or failed validation. The VALIDATION_MESSAGE element will also contain a SCHEMA_FILE_PATH element. The SCHEMA_FILE_PATH element will be present regardless of if the request XML Data Packet passed or failed validation and will include the full URI for the XSD that was used for validating the request XML Data Packet. In addition, if the RESULT element contains “Passed” then only the RESULT and SCHEMA_FILE_PATH elements will be present as child elements of the VALIDATION_MESSAGE. However, if the request XML Data Packet fails validation, and the RESULT element contains a value of “Failed”, then the VALIDATION_MESSAGE will contain one or more VALIDATION_ERROR elements. The VALIDATION_ERROR element will contain SEVERITY and MESSAGE elements that will detail exactly what failed in the request XML Data Packet as well as LINE_NUMBER and LINE_POSITION attributes that will define exactly where the validation error occurred.

The host system should always check each response to make sure the RESULT child element of the VALIDATION_MESSAGE is set to “Passed”. If it is not then there are validation errors and the transaction was not processed. The host system will have to correct any validation errors outlined in the VALIDATION_ERROR element(s) and then resubmit the request XML Data Packet.

11.0 Data Packet – XML Specification

The data packet is an XML string sent using the AuthGatewayCertification, ProcessSingleCheck, and ProcessSingleCheckWithToken web methods. The XML data packet must conform to the XSD specified in the Terminal Settings. The XML Template provided in the Terminal Settings can be used as a basis to create the Data Packet.

NOTE: *Methods with Token will operate the same as those without tokens. Tokens are used in place of Account Type, Routing Number, and Account Number.*

11.1 Authorization Gateway XML Data Packet Example:

This XML data packet example contains all available elements. The elements and data types that are required for a specific terminal are defined in that terminal’s XSD.

```
<?xml version=”1.0” encoding=”utf-8”?>
<AUTH_GATEWAY REQUEST_ID=”4654”>
   <TRANSACTION>
        <TRANSACTION_ID>0a4f529d-70fd-4ddb-b909-
        b5598dc07579</TRANSACTION_ID>
    <MERCHANT>
         <TERMINAL_ID>1113</TERMINAL_ID>
    </MERCHANT>
    <PACKET>
     <IDENTIFIER>A</IDENTIFIER>
     <CONTROL_CHAR>S</CONTROL_CHAR>
     <VERIFICATION_ONLY>false</VERIFICATION_ONLY>
     <ACCOUNT>
          <MICR_DATA>T490000018T 24413815O 4456</MICR_DATA>
          <ROUTING_NUMBER>490000018</ROUTING_NUMBER>
```

```
            <ACCOUNT_NUMBER>24413815</ACCOUNT_NUMBER>
            <CHECK_NUMBER>4456</CHECK_NUMBER>
            <ACCOUNT_TYPE>Checking</ACCOUNT_TYPE>
        </ACCOUNT>
        <CONSUMER>
            <FIRST_NAME>Test</FIRST_NAME>
            <LAST_NAME>Guy</LAST_NAME>
            <ADDRESS1>1001 Test Ave.</ADDRESS1>
            <ADDRESS2>#200</ADDRESS2>
            <CITY>Destin</CITY>
            <STATE>FL</STATE>
            <ZIP>32540</ZIP>
            <PHONE_NUMBER>2345678912</PHONE_NUMBER>
            <DL_STATE>FL</DL_STATE>
            <DL_NUMBER>D12346544</DL_NUMBER>
            <COURTESY_CARD_ID></COURTESY_CARD_ID>
            <IDENTITY>
                <DOB_YEAR>1961</DOB_YEAR>
            </IDENTITY>
        </CONSUMER>
        <CHECK>
            <CHECK_AMOUNT>1.25</CHECK_AMOUNT>
            <IMAGE_FRONT />
            <IMAGE_BACK />
        </CHECK>
        <CUSTOM>
            <CUSTOM1></CUSTOM1>
            <CUSTOM2></CUSTOM2>
            <CUSTOM3></CUSTOM3>
            <CUSTOM4></CUSTOM4>
        </CUSTOM>
      </PACKET>
    </TRANSACTION>
</AUTH_GATEWAY>
```

The Authorization Gateway XML data packet may contain the following elements:

- AUTH_GATEWAY: Is the parent element and contains all other elements within the Auth Gateway Request XML document.
- REQUEST_ID: Is an optional attribute that contains a unique user defined ID to identify the authorization gateway request. The Request ID will be returned in the Authorization Gateway response. It only persists with that specific connection and once the authorization is returned it is gone. The purpose of this attribute is to link the authorization request with the returned response and to provide a "lookup" value for the GetArchivedResponse method.
- TRANSACTION: Contains all of the elements for a given transaction.
- TRANSACTION_ID: Is an optional element that contains a unique user defined ID to identify the transaction. The Transaction ID will be returned in the Authorization Gateway response.
- MERCHANT: Contains all of the elements for the merchant.
- TERMINAL_ID: Contains the ID for the Terminal. The Terminal ID will be numeric value.

- PACKET: Contains all of the elements for packet.
- IDENTIFIER: Contains a value that identifies the packet being sent as an Authorization, Void, Override, or Payroll transaction. The identifier is a single alpha character. A list of identifiers follows, but valid identifiers will vary by schema. A=Authorize, R=Recurring, V=Void, F = Reversal, O=Override, P=Payroll
- NOTE: Identifier R (Recurring) is in name only. We will NOT run the transaction more than once.
- CONTROL_CHAR: Contains a value that identifies the information in the packet as being entered manually or retrieved from a check reader. Valid control characters are as follows: M=Manual, S=Swipe.
- VERIFICATION_ONLY: Contains a true or false value identifying if the transaction should be processed as verification only. NOTE: The Boolean data type in the XSD will require that true/false be all lower case.
- ACCOUNT: Contains all of the elements for a given account.
- MICR_DATA: Contains the MICR data read from a check reader. The MICR Data can be up to 200 characters including the following: 0-9, T, O, -.
- ROUTING_NUMBER: Contains the keyed in 9 digit routing number.
- ACCOUNT_NUMBER: Contains the keyed in account number. Valid account numbers should be between 3 and 18 numeric characters.
- CHECK_NUMBER: Contains the keyed in check number. Valid check numbers should be between 1 and 15 characters.
- ACCOUNT_TYPE: Contains the type of account. Valid values are Checking or Savings.
- CONSUMER: Contains all of the elements for a given consumer.
- FIRST_NAME: Contains the first name of the consumer. The First Name can be up to 100 alpha characters.
- LAST_NAME: Contains the last name of the consumer. The Last Name can be up to 100 alpha characters.
- ADDRESS1: Contains the first line of the consumer's address. The Address1 can be up to 200 alpha-numeric characters and can include the following: # , - , : , ;
- ADDRESS2: Contains the second line of the consumer's address. The Address2 can be up to 200 alpha-numeric characters and can include the following: # , - , : , ;
- CITY: Contains the city of the consumer's address. The City can be up to 50 alpha characters.
- STATE: Contains the state or province of the consumer's address. Valid state and province codes can be found here.
- ZIP: Contains the zip code of the consumer's address.
- PHONE_NUMBER: Contains the consumer's contact phone number. The phone number is expected as a 10 digit number without a – or ().
- DL_STATE: Contains the consumer's driver's license state or province code. Valid state and province codes can be found here.
- DL_NUMBER: Contains the consumer's driver's license number. The driver's license number can be up to 50 alpha-numeric characters.
- COURTESY_CARD_ID: Contains the consumer's courtesy card ID. The Courtesy Card ID can be up to 50 alpha-numeric characters. This element will be used if a merchant is setup with driver's license required, a courtesy card number may be substituted. This is a number generated by the merchant for the specific customer. This is most common in check cashing environments.

- IDENTITY: Contains all of the possible elements available for identifying the consumer. The IDENTITY element can only contain one child element. If the XML data packet template provided in the terminal settings contains more than one child element, than one element must be populated with a value and the other elements removed from the XML data packet prior to submitting it to the Authorization Gateway.
- SSN4: Contains the last four digits of the consumer's social security number. The SSN4 must be 4 numeric characters.
- DOB_YEAR: Contains the date of birth of the consumer. The date of birth must be 4 numeric characters begin with either 19 or 20.
- CHECK: Contains all of the elements for the check.
- CHECK_AMOUNT: Contains the amount of the check and should be between $0.01 and $999999.99.
- IMAGE_FRONT: Contains the image data for the check front. Image data must be base64.
- SIZE: The size attribute contains the image size in bytes. The size can be expressed as a decimal.
- TYPE: The type attribute contains the content type of the image. Valid TYPE values are "tiff".
- IMAGE_BACK: Contains the image data for the check back. Image data must be base64.
- SIZE: The size attribute contains the image size in bytes. The size can be expressed as a decimal.
- TYPE: The type attribute contains the content type of the image. Valid TYPE values are "tiff".
- CUSTOM1- CUSTOM4: These are optional elements that can contain up to 50 alpha numeric characters. We will return this in reporting.

11.2 Authorization Gateway XML Data Packet with IPVerification Example:

This XML data packet example contains all available elements. The elements and data types that are required for a specific terminal are defined in that terminal's XSD.

```
<?xml version="1.0" encoding="utf-8"?>
<AUTH_GATEWAY REQUEST_ID="798112">
    <TRANSACTION>
        <TRANSACTION_ID>0a4f529d-70fd-4ddb-b909-
        5598dc07579</TRANSACTION_ID>
        <MERCHANT>
              <TERMINAL_ID>2321</TERMINAL_ID>
        </MERCHANT>
        <PACKET>
              <IDENTIFIER>A</IDENTIFIER>
              <ACCOUNT>
                    <ROUTING_NUMBER>490000018</ROUTING_NUMBER>
                    <ACCOUNT_NUMBER>24413815</ACCOUNT_NUMBER>
                    <ACCOUNT_TYPE>Checking</ACCOUNT_TYPE>
              </ACCOUNT>
              <CONSUMER>
                    <FIRST_NAME>Test</FIRST_NAME>
                    <LAST_NAME>Guy</LAST_NAME>
                    <ADDRESS1>1001 Test Drive</ADDRESS1>
                    <ADDRESS2></ADDRESS2>
                    <CITY>Destin</CITY>
                    <STATE>FL</STATE>
                    <ZIP>32540</ZIP>
```

```
                <PHONE_NUMBER>8001231456</PHONE_NUMBER>
                <DL_STATE>FL</DL_STATE>
                <DL_NUMBER>D12345678910</DL_NUMBER>
                <COURTESY_CARD_ID />
                <IDENTITY>
                        <SSN4>1234</SSN4>
                </IDENTITY>
                <IP_ADDRESS>66.210.223.130</IP_ADDRESS>
            </CONSUMER>
            <CHECK>
                <CHECK_AMOUNT>1.25</CHECK_AMOUNT>
            </CHECK>
            <CUSTOM>
                <CUSTOM1></CUSTOM1>
                <CUSTOM2></CUSTOM2>
                <CUSTOM3></CUSTOM3>
                <CUSTOM4></CUSTOM4>
            </CUSTOM>
        </PACKET>
</TRANSACTION>
```

The Authorization Gateway XML data packet may contain the following elements:

- AUTH_GATEWAY: Is the parent element and contains all other elements within the Terminal Settings XML document.
- REQUEST_ID: Is an optional attribute that contains a unique user defined ID to identify the authorization gateway request. The Request ID will be returned in the Authorization Gateway response.
- TRANSACTION: Contains all of the elements for a given transaction.
- TRANSACTION_ID: Is an optional element that contains a unique user defined ID to identify the transaction. The Transaction ID will be returned in the Authorization Gateway response.
- MERCHANT: Contains all of the elements for the merchant.
- TERMINAL_ID: Contains the ID for the Terminal. The Terminal ID will be numeric value.
- PACKET: Contains all of the elements for packet.
- IDENTIFIER: Contains a value that identifies the packet being sent as an Authorization, Void, Override, or Payroll transaction. The identifier is a single alpha character. A list of identifiers follows, but valid identifiers will vary by schema. A=Authorize, V=Void, F = Reversal, O=Override
- ACCOUNT: Contains all of the elements for a given account.
- ROUTING_NUMBER: Contains the keyed in 9 digit routing number.
- ACCOUNT_NUMBER: Contains the keyed in account number. Valid account numbers should be between 3 and 18 numeric characters.
- CONSUMER: Contains all of the elements for a given consumer.
- FIRST_NAME: Contains the first name of the consumer. The First Name can be up to 100 alpha characters.
- LAST_NAME: Contains the last name of the consumer. The Last Name can be up to 100 alpha characters.
- ADDRESS1: Contains the first line of the consumer's address. The Address1 can be up to 200 alpha-numeric characters and can include the following: # , - , : , ;

- ADDRESS2: Contains the second line of the consumer's address. The Address2 can be up to 200 alpha-numeric characters and can include the following: # , - , : , ;
- CITY: Contains the city of the consumer's address. The City can be up to 50 alpha characters.
- STATE: Contains the state or province of the consumer's address. Valid state and province codes can be found here.
- ZIP: Contains the zip code of the consumer's address.
- PHONE_NUMBER: Contains the consumer's contact phone number. The phone number is expected as a 10 digit number without a – or ().
- DL_STATE: Contains the consumer's driver's license state or province code. Valid state and province codes can be found here.
- DL_NUMBER: Contains the consumer's driver's license number. The driver's license number can be up to 50 alpha-numeric characters.
- COURTSEY_CARD_ID: Contains the consumer's courtesy card ID. The Courtesy Card ID can be up to 50 alpha-numeric characters. This element will be used if a merchant is setup with driver's license required, a courtesy card number may be substituted. This is a number generated by the merchant for the specific customer. This is most common in check cashing environments.
- IDENTITY: Contains all of the possible elements available for identifying the consumer. The IDENTITY element can only contain one child element. If the XML data packet template provided in the terminal settings contains more than one child element, than one element must be populated with a value and the other elements removed from the XML data packet prior to submitting it to the Authorization Gateway.
- SSN4: Contains the last four digits of the consumer's social security number. The SSN4 must be 4 numeric characters.
- DOB_YEAR: Contains the date of birth of the consumer. The date of birth must be 4 numeric characters begin with either 19 or 20.
- IP_ADDRESS: Contains the IP address of the consumer.
- CHECK: Contains all of the elements for the check.
- CHECK_AMOUNT: Contains the amount of the check and should be between $0.01 and $999999.99.
- CUSTOM1- CUSTOM4: These are optional elements that can contain up to 50 alpha numeric characters. We will return this in reporting.

11.3 Authorization Gateway XML Data Packet with Token Example:

This XML data packet example contains all available elements. The elements and data types that are required for a specific terminal are defined in that terminal's XSD.

```
<?xml version="1.0" encoding="utf-8"?>
<AUTH_GATEWAY REQUEST_ID="4654">
  <TRANSACTION>
      <TRANSACTION_ID>0a4f529d-70fd-4ddb-
        b909b5598dc07579</TRANSACTION_ID>
  <MERCHANT>
      <TERMINAL_ID>1113</TERMINAL_ID>
  </MERCHANT>
  <PACKET>
      <IDENTIFIER>A</IDENTIFIER>
      <CONTROL_CHAR>S</CONTROL_CHAR>
```

```
        <VERIFICATION_ONLY>false</VERIFICATION_ONLY>
        <ACCOUNT>
             <TOKEN>05944FB3E1DA4663868455AF630F45BE</TOKEN>
             <CHECK_NUMBER>4456</CHECK_NUMBER>
        </ACCOUNT>
        <CONSUMER>
             <FIRST_NAME>Test</FIRST_NAME>
             <LAST_NAME>Guy</LAST_NAME>
             <ADDRESS1>1001 Test Ave.</ADDRESS1>
             <ADDRESS2>#200</ADDRESS2>
             <CITY>Destin</CITY>
             <STATE>FL</STATE>
             <ZIP>32540</ZIP>
             <PHONE_NUMBER>2345678912</PHONE_NUMBER>
             <DL_STATE>FL</DL_STATE>
             <DL_NUMBER>D12346544</DL_NUMBER>
             <COURTESY_CARD_ID></COURTESY_CARD_ID>
             <IDENTITY>
                  <DOB_YEAR>1961</DOB_YEAR>
             </IDENTITY>
        </CONSUMER>
        <CHECK>
              <CHECK_AMOUNT>1.25</CHECK_AMOUNT>
              <IMAGE_FRONT />
              <IMAGE_BACK />
        </CHECK>
        <CUSTOM>
              <CUSTOM1></CUSTOM1>
              <CUSTOM2></CUSTOM2>
              <CUSTOM3></CUSTOM3>
              <CUSTOM4></CUSTOM4>
        </CUSTOM>
    </PACKET>
  </TRANSACTION>
</AUTH_GATEWAY>
```

The Authorization Gateway XML data packet may contain the following elements:

- AUTH_GATEWAY: Is the parent element and contains all other elements within the Terminal Settings XML document.
- REQUEST_ID: Is an optional attribute that contains a unique user defined ID to identify the authorization gateway request. The Request ID will be returned in the Authorization Gateway response.
- TRANSACTION: Contains all of the elements for a given transaction.
- TRANSACTION_ID: Is an optional element that contains a unique user defined ID to identify the transaction. The Transaction ID will be returned in the Authorization Gateway response.
- MERCHANT: Contains all of the elements for the merchant.
- TERMINAL_ID: Contains the ID for the Terminal. The Terminal ID will be numeric value.
- PACKET: Contains all of the elements for packet.

- IDENTIFIER: Contains a value that identifies the packet being sent as an Authorization, Void, Override, or Payroll transaction. The identifier is a single alpha character. A list of identifiers follows, but valid identifiers will vary by schema. A=Authorize, R=Recurring, V=Void, F = Reversal, O=Override, P=Payroll
- NOTE: Identifier R (Recurring) is in name only. We will NOT run the transaction more than once.
- CONTROL_CHAR: Contains a value that identifies the information in the packet as being entered manually or retrieved from a check reader. Valid control characters are as follows: M=Manual, S=Swipe.
- VERIFICATION_ONLY: Contains a true or false value identifying if the transaction should be processed as verification only. NOTE: The Boolean data type in the XSD will require that true/false be all lower case.
- ACCOUNT: Contains all of the elements for a given account.
- TOKEN: Contains a 32 character alphanumeric value all uppercase. NOTE: This token replaces the account type, routing number, and account number.
- CHECK_NUMBER: Contains the keyed in check number. Valid check numbers should be between 1 and 15 characters.
- CONSUMER: Contains all of the elements for a given consumer.
- FIRST_NAME: Contains the first name of the consumer. The First Name can be up to 100 alpha characters.
- LAST_NAME: Contains the last name of the consumer. The Last Name can be up to 100 alpha characters.
- ADDRESS1: Contains the first line of the consumer's address. The Address1 can be up to 200 alpha-numeric characters and can include the following: # , - , : , ;
- ADDRESS2: Contains the second line of the consumer's address. The Address2 can be up to 200 alpha-numeric characters and can include the following: # , - , : , ;
- CITY: Contains the city of the consumer's address. The City can be up to 50 alpha characters.
- STATE: Contains the state or province of the consumer's address. Valid state and province codes can be found here.
- ZIP: Contains the zip code of the consumer's address.
- PHONE_NUMBER: Contains the consumer's contact phone number. The phone number is expected as a 10 digit number without a – or ().
- DL_STATE: Contains the consumer's driver's license state or province code. Valid state and province codes can be found here.
- DL_NUMBER: Contains the consumer's driver's license number. The driver's license number can be up to 50 alpha-numeric characters.
- COURTSEY_CARD_ID: Contains the consumer's courtesy card ID. The Courtesy Card ID can be up to 50 alpha-numeric characters. This element will be used if a merchant is setup with driver's license required, a courtesy card number may be substituted. This is a number generated by the merchant for the specific customer. This is most common in check cashing environments.
- IDENTITY: Contains all of the possible elements available for identifying the consumer. The IDENTITY element can only contain one child element. If the XML data packet template provided in the terminal settings contains more than one child element, than one element must be populated with a value and the other elements removed from the XML data packet prior to submitting it to the Authorization Gateway.
- SSN4: Contains the last four digits of the consumer's social security number. The SSN4 must be 4 numeric characters.

- DOB_YEAR: Contains the date of birth of the consumer. The date of birth must be 4 numeric characters begin with either 19 or 20.
- CHECK: Contains all of the elements for the check.
- CHECK_AMOUNT: Contains the amount of the check and should be between $0.01 and $999999.99.
- IMAGE_FRONT: Contains the image data for the check front. Image data must be base64.
- SIZE: The size attribute contains the image size in bytes. The size can be expressed as a decimal.
- TYPE: The type attribute contains the content type of the image. Valid TYPE values are "tiff".
- IMAGE_BACK: Contains the image data for the check back. Image data must be base64.
- SIZE: The size attribute contains the image size in bytes. The size can be expressed as a decimal.
- TYPE: The type attribute contains the content type of the image. Valid TYPE values are "tiff".
- CUSTOM1- CUSTOM4: These are optional elements that can contain up to 50 alpha numeric characters. We will return this in reporting.

12.0 How to determine which XML Template to Use

The XML data packet can be built from scratch by the web service consumer or one of the available XML templates can be used to build the XML data packet prior to submitting the data packet to the Authorization Gateway. The URI for the XML data packet for a given terminal can be retrieved from the Terminal Settings, but can also be determined by using the criteria below.

The root path for all XML Templates is https://demo.eftchecks.com/webserivces/schemas/ followed by the SEC Code, "/Templates/", and the XML Template name. The XML Template is determined by the following criteria:

- If the Terminal requires the Driver's License Information.
- If the Terminal is configured for Check Verification.
- If the Terminal is configured for Identity Verification.

A matrix of the available XML Templates for each SEC code can be found below. Each grid contains the name of the XML Template, based on the XML Templates determining criteria, and a link to the actual XML Template.

The grid also includes the Terminal IDs that can be used for testing and certifying the XML data packet that can be built from the provided XML Template. The Terminal ID will be different for guaranteed transactions and Non-guaranteed transactions. Guaranteed terminals are numbered 1xxx, and Non-guaranteed terminals are numbered 2xxx.

There are also published example XML data packets that contain example data. These published examples are only available for the PPD SEC code.
https://demo.eftchecks.com/webservices/schemas/ppd/examples/CheckVerificationIdentityVerificationDLOptional.xml

12.1 Standard XML Templates

A matrix of the available XML Templates for each SEC code can be found below. The grid contains the name of the XML Template, based on the XML Templates determining criteria, and a link to the actual XML Template.

12.1.1 PPD XML Templates

(Root path: https://demo.eftchecks.com/webservices/schemas/ppd/templates)

Template	DL Required	Verify Check	Verify ID	Certification Terminal ID (Guar/Non)
CheckNoVerificationDLOptional.xml				1010 / 2010
CheckNoVerificationDLRequired.xml	X			1011 / 2011
CheckVerificationIdentityVerificationDLOptional.xml		X	X	1012 / 2012
CheckVerificationIdentityVerificationDLRequired.xml	X	X	X	1013 / 2013
CheckVerificationOnlyDLOptional.xml		X		1014 / 2014
CheckVerificationOnlyDLRequired.xml	X	X		1015 / 2015
IdentityVerificationOnlyDLOptional.xml			X	1016 / 2016
IdentityVerificationOnlyDLRequired.xml	X		X	1017 / 2017

12.1.2 CCD XML Templates

(Root path: https://demo.eftchecks.com/webservices/schemas/ccd//templates)

Template	DL Required	Verify Check	Verify ID	Certification Terminal ID (Guar/Non)
CheckNoVerificationDLOptional.xml				1710 / 2710
CheckNoVerificationDLRequired.xml	X			1711 / 2711
CheckVerificationIdentityVerificationDLOptional.xml		X	X	1712 / 2712
CheckVerificationIdentityVerificationDLRequired.xml	X	X	X	1713 / 2713
CheckVerificationOnlyDLOptional.xml		X		1714 / 2714
CheckVerificationOnlyDLRequired.xml	X	X		1715 / 2715
IdentityVerificationOnlyDLOptional.xml			X	1716 / 2716
IdentityVerificationOnlyDLRequired.xml	X		X	1717 / 2717

12.1.3 WEB XML Templates

(Root path: https://demo.eftchecks.com/webservices/schemas/web/templates)

Template	DL Required	Verify Check	Verify ID	Certification Terminal ID
CheckNoVerificationDLOptional.xml				2310
CheckNoVerificationDLRequired.xml	X			2311
CheckVerificationIdentityVerificationDLOptional.xml		X	X	2312
CheckVerificationIdentityVerificationDLRequired.xml	X	X	X	2313
CheckVerificationOnlyDLOptional.xml		X		2314
CheckVerificationOnlyDLRequired.xml	X	X		2315
IdentityVerificationOnlyDLOptional.xml			X	2316
IdentityVerificationOnlyDLRequired.xml	X		X	2317

12.1.4 TEL XML Templates

(Root path: https://demo.eftchecks.com/webservices/schemas/tel/templates)

Template	DL Required	Verify Check	Verify ID	Certification Terminal ID (Guar/Non)
CheckNoVerificationDLOptional.xml				1210 / 2210
CheckNoVerificationDLRequired.xml	X			1211 / 2211
CheckVerificationIdentityVerificationDLOptional.xml		X	X	1212 / 2212
CheckVerificationIdentityVerificationDLRequired.xml	X	X	X	1213 / 2213
CheckVerificationOnlyDLOptional.xml		X		1214 / 2214
CheckVerificationOnlyDLRequired.xml	X	X		1215 / 2215
IdentityVerificationOnlyDLOptional.xml			X	1216 / 2216
IdentityVerificationOnlyDLRequired.xml	X		X	1217 / 2217

12.1.5 POP XML Templates

(Root path: https://demo.eftchecks.com/webservices/schemas/pop/templates)

Template	DL Required	Verify Check	Verify ID	Certification Terminal ID
CheckNoVerificationDLOptional.xml				1110
CheckNoVerificationDLRequired.xml	X			1111
CheckVerificationIdentityVerificationDLOptional.xml		X	X	1112
CheckVerificationIdentityVerificationDLRequired.xml	X	X	X	1113
CheckVerificationOnlyDLOptional.xml		X		1114
CheckVerificationOnlyDLRequired.xml	X	X		1115
IdentityVerificationOnlyDLOptional.xml			X	1116
IdentityVerificationOnlyDLRequired.xml	X		X	1117

12.1.6 Check21 XML Templates

(Root path: https://demo.eftchecks.com/webservices/schemas/c21/templates)

Template	DL Required	Verify Check	Verify ID	Certification Terminal ID
CheckNoVerificationDLOptional.xml				1610
CheckNoVerificationDLRequired.xml	X			1611
CheckVerificationIdentityVerificationDLOptional.xml		X	X	1612
CheckVerificationIdentityVerificationDLRequired.xml	X	X	X	1613
CheckVerificationOnlyDLOptional.xml		X		1614
CheckVerificationOnlyDLRequired.xml	X	X		1615
IdentityVerificationOnlyDLOptional.xml			X	1616
IdentityVerificationOnlyDLRequired.xml	X		X	1617

12.2 XML Templates when using IPVerification

A matrix of the available XML IPVerification Templates for WEB SEC code can be found below. The grid contains the name of the XML Template, based on the XML Templates determining criteria, and a link to the actual XML Template.

12.2.1 WEB XML Templates

(Root path: https://demo.eftchecks.com/webservices/schemas/web/templates)

Template	DL Required	Verify Check	Verify ID	Certification Terminal ID
CheckNoVerificationDLOptionalwithIP.xml				2318
CheckNoVerificationDLRequiredwithIP.xml	X			2319
CheckVerificationIdentityVerificationDLOptionalwithIP.xml		X	X	2320
CheckVerificationIdentityVerificationDLRequiredwithIP.xml	X	X	X	2321
CheckVerificationOnlyDLOptionalwithIP.xml		X		2322
CheckVerificationOnlyDLRequiredwithIP.xml	X	X		2323
IdentityVerificationOnlyDLOptionalwithIP.xml			X	2324
IdentityVerificationOnlyDLRequiredwithIP.xml	X		X	2325

12.3 XML Templates when using Tokens

A matrix of the available XML Templates when using tokens for each SEC code can be found below. Each grid contains the name of the XML Template, based on the XML Templates determining criteria, and a link to the actual XML Template.

12.3.1 PPD XML Templates

(Root path: https://demo.eftchecks.com/webservices/schemas/ppd/templates)

Template	DL Required	Verify Check	Verify ID	Certification Terminal ID (Guar/Non)
CheckNoVerificationDLWithTokenOptional.xml				1010 / 2010
CheckNoVerificationDLWithTokenRequired.xml	X			1011 / 2011
CheckVerificationIdentityVerificationDLWithTokenOptional.xml		X	X	1012 / 2012
CheckVerificationIdentityVerificationDLWithTokenRequired.xml	X	X	X	1013 / 2013
CheckVerificationOnlyDLWithTokenOptional.xml		X		1014 / 2014
CheckVerificationOnlyDLWithTokenRequired.xml	X	X		1015 / 2015
IdentityVerificationOnlyDLWithTokenOptional.xml			X	1016 / 2016
IdentityVerificationOnlyDLWithTokenRequired.xml	X		X	1017 / 2017

12.3.1 CCD XML Templates

(Root path: https://demo.eftchecks.com/webservices/schemas/ccd//templates)

Template	DL Required	Verify Check	Verify ID	Certification Terminal ID (Guar/Non)
CheckNoVerificationDLWithTokenOptional.xml				1710 / 2710
CheckNoVerificationDLWithTokenRequired.xml	X			1711 / 2711
CheckVerificationIdentityVerificationDLWithTokenOptional.xml		X	X	1712 / 2712
CheckVerificationIdentityVerificationDLWithTokenRequired.xml	X	X	X	1713 / 2713
CheckVerificationOnlyDLWithTokenOptional.xml		X		1714 / 2714
CheckVerificationOnlyDLWithTokenRequired.xml	X	X		1715 / 2715
IdentityVerificationOnlyDLWithTokenOptional.xml			X	1716 / 2716
IdentityVerificationOnlyDLWithTokenRequired.xml	X		X	1717 / 2717

12.3.1 WEB XML Templates

(Root path: https://demo.eftchecks.com/webservices/schemas/web/templates)

Template	DL Required	Verify Check	Verify ID	Certification Terminal ID
CheckNoVerificationDLWithTokenOptional.xml				2310
CheckNoVerificationDLWithTokenRequired.xml	X			2311
CheckVerificationIdentityVerificationDLWithTokenOptional.xml		X	X	2312
CheckVerificationIdentityVerificationDLWithTokenRequired.xml	X	X	X	2313
CheckVerificationOnlyDLWithTokenOptional.xml		X		2314
CheckVerificationOnlyDLWithTokenRequired.xml	X	X		2315
IdentityVerificationOnlyDLWithTokenOptional.xml			X	2316
IdentityVerificationOnlyDLWithTokenRequired.xml	X		X	2317

12.3.1 TEL XML Templates

(Root path: https://demo.eftchecks.com/webservices/schemas/tel/templates)

Template	DL Required	Verify Check	Verify ID	Certification Terminal ID (Guar/Non)
CheckNoVerificationDLWithTokenOptional.xml				1210 / 2210
CheckNoVerificationDLWithTokenRequired.xml	X			1211 / 2211
CheckVerificationIdentityVerificationDLWithTokenOptional.xml		X	X	1212 / 2212
CheckVerificationIdentityVerificationDLWithTokenRequired.xml	X	X	X	1213 / 2213
CheckVerificationOnlyDLWithTokenOptional.xml		X		1214 / 2214
CheckVerificationOnlyDLWithTokenRequired.xml	X	X		1215 / 2215
IdentityVerificationOnlyDLWithTokenOptional.xml			X	1216 / 2216
IdentityVerificationOnlyDLWithTokenRequired.xml	X		X	1217 / 2217

12.3.1 POP XML Templates

(Root path: https://demo.eftchecks.com/webservices/schemas/pop/templates)

Template	DL Required	Verify Check	Verify ID	Certification Terminal ID
CheckNoVerificationDLWithTokenOptional.xml				1110
CheckNoVerificationDLWithTokenRequired.xml	X			1111
CheckVerificationIdentityVerificationDLWithTokenOptional.xml		X	X	1112
CheckVerificationIdentityVerificationDLWithTokenRequired.xml	X	X	X	1113
CheckVerificationOnlyDLWithTokenOptional.xml		X		1114
CheckVerificationOnlyDLWithTokenRequired.xml	X	X		1115
IdentityVerificationOnlyDLWithTokenOptional.xml			X	1116
IdentityVerificationOnlyDLWithTokenRequired.xml	X		X	1117

12.3.1 Check21 XML Templates

(Root path: https://demo.eftchecks.com/webservices/schemas/c21/templates)

Template	DL Required	Verify Check	Verify ID	Certification Terminal ID
CheckNoVerificationDLWithTokenOptional.xml				1610
CheckNoVerificationDLWithTokenRequired.xml	X			1611

CheckVerificationIdentityVerificationDLWithTokenOptional.xml		X	X	1612
CheckVerificationIdentityVerificationDLWithTokenRequired.xml	X	X	X	1613
CheckVerificationOnlyDLWithTokenOptional.xml		X		1614
CheckVerificationOnlyDLWithTokenRequired.xml	X	X		1615
IdentityVerificationOnlyDLWithTokenOptional.xml			X	1616
IdentityVerificationOnlyDLWithTokenRequired.xml	X		X	1617

13.0 How to determine which XSD to Use

The XSD that will be used can be retrieved from the Terminal Settings, but can also be determined by using the criteria below.

The root path for all XSDs is https://demo.eftchecks.com/webservices/pub_schemas followed by the SEC Code and Schema Name. The Schema Name is determined by the following criteria:

- If the Terminal requires the Driver's License Information.
- If the Terminal is configured for Check Verification.
- If the Terminal is configured for Identity Verification.
- For PPD and CCD entries, If the Terminal is configured to allow Credit entries

A matrix of the available XSDs for each SEC code can be found below. Each grid contains the name of the schema, based on the schemas determining criteria, and a link to the actual schema. The grid also includes the Terminal IDs that can be used for testing and certifying against the provided schema.

An example XSD file path for a PPD terminal that does *not* require the driver's license information, *is* setup for check verification, and *is* setup for identity verification, and *does not* allow credits would be as follows:
https://demo.eftchecks.com/webservices/schemas/ppd/CheckVerificationIdentityVerificationDLOptional.xsd

13.1 Standard XSD Schemas

A matrix of the available XSDs can be found below. Each grid contains the name of the schema, based on the schemas determining criteria, and a link to the actual schema. The grid also includes the Terminal IDs that can be used for testing and certifying against the provided schema.

13.1.1 PPD Schemas – Guaranteed

(Root path: https://demo.eftchecks.com/webservices/pub_schemas/ppd/)

Template	DL Required	Verify Check	Verify ID	Certification Terminal ID
Debit Only Transactions				
CheckNoVerificationDLOptional.xsd				1010
CheckNoVerificationDLRequired.xsd	X			1011
CheckVerificationIdentityVerificationDLOptional.xsd		X	X	1012
CheckVerificationIdentityVerificationDLRequired.xsd	X	X	X	1013
CheckVerificationOnlyDLOptional.xsd		X		1014
CheckVerificationOnlyDLRequired.xsd	X	X		1015
IdentityVerificationOnlyDLOptional.xsd			X	1016
IdentityVerificationOnlyDLRequired.xsd	X		X	1017
Credit & Debit Transactions				
CreditCheckNoVerificationDLOptional.xsd				1810
CreditCheckNoVerificationDLRequired.xsd	X			1811
CreditCheckVerificationIdentityVerificationDLOptional.xsd		X	X	1812
CreditCheckVerificationIdentityVerificationDLRequired.xsd	X	X	X	1813
CreditCheckVerificationOnlyDLOptional.xsd		X		1814
CreditCheckVerificationOnlyDLRequired.xsd	X	X		1815
CreditIdentityVerificationOnlyDLOptional.xsd			X	1816
CreditIdentityVerificationOnlyDLRequired.xsd	X		X	1817

13.1.2 PPD Schemas – Non-Guaranteed

(Root path: https://demo.eftchecks.com/webservices/pub_schemas/ppd/)

Template	DL Required	Verify Check	Verify ID	Certification Terminal ID
Debit Only Transactions				
Ng_CheckNoVerificationDLOptional.xsd				2010
Ng_CheckNoVerificationDLRequired.xsd	X			2011
Ng_CheckVerificationIdentityVerificationDLOptional.xsd		X	X	2012
Ng_CheckVerificationIdentityVerificationDLRequired.xsd	X	X	X	2013
Ng_CheckVerificationOnlyDLOptional.xsd		X		2014
Ng_CheckVerificationOnlyDLRequired.xsd	X	X		2015
Ng_IdentityVerificationOnlyDLOptional.xsd			X	2016

Ng_IdentityVerificationOnlyDLRequired.xsd	X		X	2017
Credit & Debit Transactions				
Ng_CreditCheckNoVerificationDLOptional.xsd				2810
Ng_CreditCheckNoVerificationDLRequired.xsd	X			2811
Ng_CreditCheckVerificationIdentityVerificationDLOptional.xsd		X	X	2812
Ng_CreditCheckVerificationIdentityVerificationDLRequired.xsd	X	X	X	2813
Ng_CreditCheckVerificationOnlyDLOptional.xsd		X		2814
Ng_CreditCheckVerificationOnlyDLRequired.xsd	X	X		2815
Ng_CreditIdentityVerificationOnlyDLOptional.xsd			X	2816
Ng_CreditIdentityVerificationOnlyDLRequired.xsd	X		X	2817

13.1.3 CCD Schemas – Guaranteed

(Root path: https://demo.eftchecks.com/webservices/pub_schemas/ccd/)

Template	DL Required	Verify Check	Verify ID	Certification Terminal ID
Debit Only Transactions				
CheckNoVerificationDLOptional.xsd				1710
CheckNoVerificationDLRequired.xsd	X			1711
CheckVerificationIdentityVerificationDLOptional.xsd		X	X	1712
CheckVerificationIdentityVerificationDLRequired.xsd	X	X	X	1713
CheckVerificationOnlyDLOptional.xsd		X		1714
CheckVerificationOnlyDLRequired.xsd	X	X		1715
IdentityVerificationOnlyDLOptional.xsd			X	1716
IdentityVerificationOnlyDLRequired.xsd	X		X	1717
Credit & Debit Transactions				
CreditCheckNoVerificationDLOptional.xsd				1910
CreditCheckNoVerificationDLRequired.xsd	X			1911
CreditCheckVerificationIdentityVerificationDLOptional.xsd		X	X	1912
CreditCheckVerificationIdentityVerificationDLRequired.xsd	X	X	X	1913
CreditCheckVerificationOnlyDLOptional.xsd		X		1914
CreditCheckVerificationOnlyDLRequired.xsd	X	X		1915
CreditIdentityVerificationOnlyDLOptional.xsd			X	1916
CreditIdentityVerificationOnlyDLRequired.xsd	X		X	1917

13.1.4 CCD Schemas – Non-Guaranteed

(Root path: https://demo.eftchecks.com/webservices/pub_schemas/ccd/)

Template	DL Required	Verify Check	Verify ID	Certification Terminal ID
Debit Only Transactions				
Ng_CheckNoVerificationDLOptional.xsd				2710
Ng_CheckNoVerificationDLRequired.xsd	X			2711
Ng_CheckVerificationIdentityVerificationDLOptional.xsd		X	X	2712
Ng_CheckVerificationIdentityVerificationDLRequired.xsd	X	X	X	2713
Ng_CheckVerificationOnlyDLOptional.xsd		X		2714
Ng_CheckVerificationOnlyDLRequired.xsd	X	X		2715
Ng_IdentityVerificationOnlyDLOptional.xsd			X	2716
Ng_IdentityVerificationOnlyDLRequired.xsd	X		X	2717
Credit & Debit Transactions				
Ng_CreditCheckNoVerificationDLOptional.xsd				2910
Ng_CreditCheckNoVerificationDLRequired.xsd	X			2911
Ng_CreditCheckVerificationIdentityVerificationDLOptional.xsd		X	X	2912
Ng_CreditCheckVerificationIdentityVerificationDLRequired.xsd	X	X	X	2913
Ng_CreditCheckVerificationOnlyDLOptional.xsd		X		2914
Ng_CreditCheckVerificationOnlyDLRequired.xsd	X	X		2915
Ng_CreditIdentityVerificationOnlyDLOptional.xsd			X	2916
Ng_CreditIdentityVerificationOnlyDLRequired.xsd	X		X	2917

13.1.5 WEB Schemas

(Root path: https://demo.eftchecks.com/webservices/pub_schemas/web/)

Template	DL Required	Verify Check	Verify ID	Certification Terminal ID
Ng_CheckNoVerificationDLOptional.xsd				2310
Ng_CheckNoVerificationDLRequired.xsd	X			2311
Ng_CheckVerificationIdentityVerificationDLOptional.xsd		X	X	2312
Ng_CheckVerificationIdentityVerificationDLRequired.xsd	X	X	X	2313

Ng_CheckVerificationOnlyDLOptional.xsd		X		2314
Ng_CheckVerificationOnlyDLRequired.xsd	X	X		2315
Ng_IdentityVerificationOnlyDLOptional.xsd			X	2316
Ng_IdentityVerificationOnlyDLRequired.xsd	X		X	2317

13.1.6 TEL Schemas – Guaranteed

(Root path: https://demo.eftchecks.com/webservices/pub_schemas/tel/)

Template	DL Required	Verify Check	Verify ID	Certification Terminal ID
CheckNoVerificationDLOptional.xsd				1210
CheckNoVerificationDLRequired.xsd	X			1211
CheckVerificationIdentityVerificationDLOptional.xsd		X	X	1212
CheckVerificationIdentityVerificationDLRequired.xsd	X	X	X	1213
CheckVerificationOnlyDLOptional.xsd		X		1214
CheckVerificationOnlyDLRequired.xsd	X	X		1215
IdentityVerificationOnlyDLOptional.xsd			X	1216
IdentityVerificationOnlyDLRequired.xsd	X		X	1217

13.1.7 TEL Schemas – Non-Guaranteed

(Root path: https://demo.eftchecks.com/webservices/pub_schemas/tel/)

Template	DL Required	Verify Check	Verify ID	Certification Terminal ID
Ng_CheckNoVerificationDLOptional.xsd				2210
Ng_CheckNoVerificationDLRequired.xsd	X			2211
Ng_CheckVerificationIdentityVerificationDLOptional.xsd		X	X	2212
Ng_CheckVerificationIdentityVerificationDLRequired.xsd	X	X	X	2213
Ng_CheckVerificationOnlyDLOptional.xsd		X		2214
Ng_CheckVerificationOnlyDLRequired.xsd	X	X		2215
Ng_IdentityVerificationOnlyDLOptional.xsd			X	2216
Ng_IdentityVerificationOnlyDLRequired.xsd	X		X	2217

13.1.8 POP Schemas

(Root path: https://demo.eftchecks.com/webservices/pub_schemas/pop/)

Template	DL Required	Verify Check	Verify ID	Certification Terminal ID
CheckNoVerificationDLOptional.xsd				1110

CheckNoVerificationDLRequired.xsd	X			1111
CheckVerificationIdentityVerificationDLOptional.xsd		X	X	1112
CheckVerificationIdentityVerificationDLRequired.xsd	X	X	X	1113
CheckVerificationOnlyDLOptional.xsd		X		1114
CheckVerificationOnlyDLRequired.xsd	X	X		1115
IdentityVerificationOnlyDLOptional.xsd			X	1116
IdentityVerificationOnlyDLRequired.xsd	X		X	1117

13.1.9 Check21 Schemas

(Root path: https://demo.eftchecks.com/webservices/pub_schemas/c21/)

Template	DL Required	Verify Check	Verify ID	Certification Terminal ID
CheckNoVerificationDLOptional.xsd				1610
CheckNoVerificationDLRequired.xsd	X			1611
CheckVerificationIdentityVerificationDLOptional.xsd		X	X	1612
CheckVerificationIdentityVerificationDLRequired.xsd	X	X	X	1613
CheckVerificationOnlyDLOptional.xsd		X		1614
CheckVerificationOnlyDLRequired.xsd	X	X		1615
IdentityVerificationOnlyDLOptional.xsd			X	1616
IdentityVerificationOnlyDLRequired.xsd	X		X	1617

13.2 XSD Schemas when using IPVerification

A matrix of the available XSDs for IPVerification can be found below. Each grid contains the name of the schema, based on the schemas determining criteria, and a link to the actual schema. The grid also includes the Terminal IDs that can be used for testing and certifying against the provided schema.

13.2.1 WEB Schemas

(Root path: https://demo.eftchecks.com/webservices/pub_schemas/web/)

Template	DL Required	Verify Check	Verify ID	Certification Terminal ID
Ng_CheckNoVerificationDLOptionalwithIP.xsd				2318
Ng_CheckNoVerificationDLRequiredwithIP.xsd	X			2319
Ng_CheckVerificationIdentityVerificationDLOptionalwithIP.xsd		X	X	2320
Ng_CheckVerificationIdentityVerificationDLRequiredwithIP.xsd	X	X	X	2321
Ng_CheckVerificationOnlyDLOptionalwithIP.xsd		X		2322
Ng_CheckVerificationOnlyDLRequiredwithIP.xsd	X	X		2323
Ng_IdentityVerificationOnlyDLOptionalwithIP.xsd			X	2324
Ng_IdentityVerificationOnlyDLRequiredwithIP.xsd	X		X	2325

14.0 Data Types

Each element in the XML data packet that is sent to the Authorization Gateway has a data type that defines the format of the data contained within the element. The Terminal's XSD defines which elements are of what data type. A list and links to the available data types is located below.

- States and Provinces
  https://demo.eftchecks.com/Webservices/schemas/types/StatesAndProvincesSimpleType.xsd
- Authorization Gateway Types
  https://demo.eftchecks.com/Webservices/schemas/types/AuthGatewayTypes.xsd
- Authorization Gateway Response Types
  https://demo.eftchecks.com/Webservices/schemas/types/AuthGatewayResponseTypes.xsd

15.0 Responses

Each web method in the Authorization Gateway will return an XML string and detail the success or failure of the submission. If the transaction is accepted (authorized) an authorization number will be returned at a minimum.

The Authorization Gateway XML response may contain the following elements:

- REQUEST_ID: Is an attribute that contains a unique user defined ID to identify the authorization gateway request. The Request ID contained in the Authorization Gateway Request is returned in the Authorization Gateway response.
- VALIDATION_MESSAGE: Contains all of the elements in the validation message.
- AUTHORIZATION_MESSAGE: Contains all of the elements in the authorization message.
  ***NOTE**: The AuthGatewayCertification web method response will not contain this element.*

15.1 Validation Message Response

The AuthGatewayCertification, ProcessSingleCheck, and ProcessSingleCheckWithToken web methods will validate that the interface is sending a data packet that conforms to its schema.

15.1.1 Validation Message Example – Success Response

```
<?xml version="1.0" encoding="utf-8"?>
<RESPONSE xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" REQUEST_ID="4654">
        <VALIDATION_MESSAGE>
            <RESULT>Passed</RESULT>
            <SCHEMA_FILE_PATH>http://demo.eftchecks.com/webservices/Schemas/WEB/CheckNoVerificationDLOptional.xsd
            </SCHEMA_FILE_PATH>
        </VALIDATION_MESSAGE>
</RESPONSE>
```

15.1.2 Validation Message Example – Failure Response

This data packet failed validation because the Driver's License Information is required by the XSD and was not provided in the data packet.

```
<?xml version="1.0" encoding="utf-8" ?>
 <RESPONSE xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" REQUEST_ID="4654">>
    <VALIDATION_MESSAGE>
   <RESULT>Failed</RESULT>
        <SCHEMA_FILE_PATH>
              http://localhost/GETI.eMagnus.WebServices/Schemas/PPD/
               CheckNoVerificationDLRequired.xsd
        </SCHEMA_FILE_PATH>
       <VALIDATION_ERROR LINE_NUMBER="1" LINE_POSITION="561" >
       <SEVERITY>Error</SEVERITY>
       <MESSAGE>
                  The 'DL_STATE' element has an invalid value according to its data
                   type. An error occurred at (1, 561).
              </MESSAGE>
      </VALIDATION_ERROR>
       <VALIDATION_ERROR LINE_NUMBER="1" LINE_POSITION="583">
       <SEVERITY>Error</SEVERITY>
       <MESSAGE>
                  The 'IDENTIFIER' element has an invalid value according to its data
                   type.
             </MESSAGE>
      </VALIDATION_ERROR>
   </VALIDATION_MESSAGE>
   </RESPONSE>
```

The Validation Message may contain the following elements and attributes:

- RESULT: Contains Passed or Failed indicating if the validation was successful or not.
- SCHEMA_FILE_PATH: Contains the Uniform Resource Identifier (URI) specifying the published XML Schema Definition (XSD) that the data packet request will be validated against.
- VALIDATION_ERROR: Contains all of the elements in the validation error.
- LINE_NUMBER: Contains the line the number where the validation error occurred.
- LINE_POSITION: Contains the line position where the validation error occurred.
- SEVERITY: Contains warning or error indicating the severity of the validation error.
- MESSAGE: Contains the complete validation error message and will include the element that failed the validation and may contain the location the validation error occurred.

15.2 Authorization Message Response

The ProcessSingleCheck web method will process a valid XML data packet and return an Authorization Message within the response. An example of the Authorization message is below.

15.2.1 Authorization Message Example

```
<?xml version="1.0" encoding="utf-8" ?>
<RESPONSE
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"xmlns:xsi="http://www.w3.org/2001/
  XMLSchema-instance" REQUEST_ID="4654">
  <VALIDATION_MESSAGE>
      <RESULT>Passed</RESULT>
      <SCHEMA_FILE_PATH>http://localhost/GETI.eMagnus.WebServices/Schemas/PPD/C
        heckVerificationIdentityVerificationDLRequired.xsd</SCHEMA_FILE_PATH>
  </VALIDATION_MESSAGE>
  <AUTHORIZATION_MESSAGE>
      <TRANSACTION_ID>0a4f529d-70fd-4ddb-b909-
        b5598dc07579</TRANSACTION_ID>
      <RESPONSE_TYPE>A</RESPONSE_TYPE>
      <RESPONSE_TYPE_TEXT>APPROVED</RESPONSE_TYPE_TEXT>
      <RESULT_CODE>0</RESULT_CODE>
      <TYPE_CODE>4096</TYPE_CODE>
      <CODE>AUTH NUM 272-172</CODE>
      <MESSAGE>APPROVAL</MESSAGE>
  </AUTHORIZATION_MESSAGE>
</RESPONSE>
```

The Authorization Message may contain the following elements:

- TRANSACTION_ID: Contains the unique user defined ID to identify the packet. The Transaction ID provided for a given transaction in the data packet is returned in the authorization message in the response.
- RESPONSE_TYPE: contains an identifier that will give your host system a general overview of the processed transaction, and the RESPONSE_TYPE_TEXT element will contain the full text description of the identifier contained in the RESPONSE_TYPE element. The RESULT_CODE should be the primary driver for determining how the host system should act in response to various responses from the Authorization Gateway, but the values in the RESPONSE_TYPE can be used to determine additional information for processing by the host system. This includes determining if a transaction was processed as Verification Only. A complete list of response types is located in the Authorization Response Types XSD.
- RESPONSE_TYPE_TEXT: Contains the full text description of the response type identifier.
- RESULT_CODE: Contains a numeric bit that indicates one or many result messages. Examples of result messages are Approved, Decline, or Unpaid Check Limit Exceeded. A complete list of result codes is located in the Authorization Response Types XSD. The host system should conduct a bit comparison of the RESULT_CODE to determine exactly how the transaction was processed and from there can determine exactly what action to take if any.
- TYPE_CODE: Contains a numeric bit that indicates one or many type messages. The host system should conduct a bit comparison of the TYPE_CODE element to determine additional detailed information about the transaction which can be provided to the user. Examples of type messages are Personal Check, Business Check, or Voided Check. A complete list of type codes is located in the Authorization Response Types XSD.

- CODE: Contains the text message with the Authorization Number if the transaction was approved or additional information if the transaction was not approved. The CODE element should be used by the host system to display and record any authorization numbers or additional information.
- MESSAGE: Contains additional text, and should be used by the host system to display and record any additional information about the transaction.

15.2.2 Process Single Certification Check – Authorization

When processing a single certification check for Authorization you will need to invoke the ProcessSingleCertificationCheck web method and set the routing number to 490000018 in the ROUTING_NUMBER element of the request XML Data Packet. You will also have to set the value of the IDENTIFER element to "R" if you are using a PPD or CCD schema or "A" for all other schemas. If the request XML Data Packet is valid then this routing number will trigger the Authorization Gateway to return a response with the following information to the host system:

- **RESPONSE_TYPE:** A
- **RESPONSE_TYPE_TEXT:** APPROVED
- **RESULT_CODE:** 0
- **TYPE_CODE:** 4096
- **CODE:** AUTH NUM 272-172
- **MESSAGE:** APPROVAL

Again, the host system should first check to make sure the RESULT child element of the VALIDATION_MESSAGE is set to "Passed". If the request XML Data Packet passed validation it was successfully processed and the above elements will be present as child elements of the AUTHORIZATION_MESSAGE element. The host system should store all of the returned data and at a minimum conduct a bit comparison of the value in the RESULT_CODE element. If the value in the RESULT_CODE is 0 then the transaction has been approved. The response Data Packet also contains a value of 4096 for the TYPE_CODE element which indicates an Internal Override which in this instance means that Authorization Gateway returned a predetermined fixed response.

15.3 Authorization Message Response with Token

The ProcessSingleCheckWithToken web method will process a valid XML data packet and return an Authorization Message within the response. An example of the Authorization message is below.

15.3.1 Authorization Message Example with Token

```
<?xml version="1.0" encoding="utf-8" ?>
<RESPONSE
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"xmlns:xsi="http://www.w3.org/2001/
  XMLSchema-instance" REQUEST_ID="4654">
  <VALIDATION_MESSAGE>
      <RESULT>Passed</RESULT>
      <SCHEMA_FILE_PATH>http://localhost/GETI.eMagnus.WebServices/Schemas/PPD/C
        heckVerificationIdentityVerificationDLRequired.xsd</SCHEMA_FILE_PATH>
  </VALIDATION_MESSAGE>
```

```
  <AUTHORIZATION_MESSAGE>
      <TRANSACTION_ID>0a4f529d-70fd-4ddb-b909-
        b5598dc07579</TRANSACTION_ID>
      <RESPONSE_TYPE>A</RESPONSE_TYPE>
      <RESPONSE_TYPE_TEXT>APPROVED</RESPONSE_TYPE_TEXT>
      <RESULT_CODE>0</RESULT_CODE>
      <TYPE_CODE>4096</TYPE_CODE>
      <CODE>AUTH NUM 272-172</CODE>
      <MESSAGE>APPROVAL</MESSAGE>
      <TOKEN>C7E057491C4A4D67B617EE512D1300AE</TOKEN>
  </AUTHORIZATION_MESSAGE>
</RESPONSE>
```

The Authorization Message may contain the following elements:

- TRANSACTION_ID: Contains the unique user defined ID to identify the packet. The Transaction ID provided for a given transaction in the data packet is returned in the authorization message in the response.
- RESPONSE_TYPE: contains an identifier that will give your host system a general overview of the processed transaction, and the RESPONSE_TYPE_TEXT element will contain the full text description of the identifier contained in the RESPONSE_TYPE element. The RESULT_CODE should be the primary driver for determining how the host system should act in response to various responses from the Authorization Gateway, but the values in the RESPONSE_TYPE can be used to determine additional information for processing by the host system. This includes determining if a transaction was processed as Verification Only. A complete list of response types is located in the Authorization Response Types XSD.
- RESPONSE_TYPE_TEXT: Contains the full text description of the response type identifier.
- RESULT_CODE: Contains a numeric bit that indicates one or many result messages. Examples of result messages are Approved, Decline, or Unpaid Check Limit Exceeded. A complete list of result codes is located in the Authorization Response Types XSD. The host system should conduct a bit comparison of the RESULT_CODE to determine exactly how the transaction was processed and from there can determine exactly what action to take if any.
- TYPE_CODE: Contains a numeric bit that indicates one or many type messages. The host system should conduct a bit comparison of the TYPE_CODE element to determine additional detailed information about the transaction which can be provided to the user. Examples of type messages are Personal Check, Business Check, or Voided Check. A complete list of type codes is located in the Authorization Response Types XSD.
- CODE: Contains the text message with the Authorization Number if the transaction was approved or additional information if the transaction was not approved. The CODE element should be used by the host system to display and record any authorization numbers or additional information.
- MESSAGE: Contains additional text, and should be used by the host system to display and record any additional information about the transaction.
- TOKEN: Contains the return Token that is used in place of the Account Type, Routing Number, and Account Number. This token can then be used for future transactions. ***NOTE:*** *This is only available when using a Token or when requesting a Token.*

15.4 Process Single Certification Check – Check Limit Exceeded

The check limit for processing single certification checks is $25. If a check amount in excess of $25 is sent to the Authorization Gateway during development or certification phases then the Authorization Gateway will return “Check Limit Exceeded – Decline”.

When processing a single certification check for Check Limit Exceeded you will need to invoke the ProcessSingleCertificationCheck web method and set the routing number to 490000018 in the ROUTING_NUMBER element of the request XML Data Packet. You will also have to set the value of the of the IDENTIFER element to “R” if you are using a PPD or CCD schema or “A” for all other schemas. Finally you will need to include a check amount larger than $25 in the CHECK_AMOUNT element. If the request XML Data Packet is valid then this will trigger the Authorization Gateway to return a response with the following information to the host system:

- **RESPONSE_TYPE:** D
- **RESPONSE_TYPE_TEXT:** DECLINED
- **RESULT_CODE:** 136
- **TYPE_CODE:** 256
- **CODE:** DECLINE CHECK LIMIT EXCEEDED
- **MESSAGE:** DECLINE CHECK LIMIT EXCEEDED

If a transaction is declined the Authorization Gateway will return an 8 in the RESULT_CODE element which indicates a Decline Message. In the returned response above the RESULT_CODE element has a value of 136. If the host system is setup to do a bit comparison of the value in the RESULT_CODE you will discover that the 136 is made of an 8 indicating a Decline Message and 128 which indicates that the transaction limit has been exceeded. The fixed decline response also contains a value of 256 in the TYPE_CODE element. This indicates that the host system was unable to determine if this was the first time the check was presented or if it is a representment.

15.5 Process Single Certification Check – Decline

When processing a single certification check for Decline you will need to invoke the ProcessSingleCertificationCheck web method and set the routing number to 490000034 in the ROUTING_NUMBER element of the request XML Data Packet. You will also have to set the value of the IDENTIFER element to “R” if you are using a PPD or CCD schema or “A” for all other schemas. If the request XML Data Packet is valid then this routing number will trigger the Authorization Gateway to return a response with the following information to the host system:

- **RESPONSE_TYPE:** D
- **RESPONSE_TYPE_TEXT:** DECLINED
- **RESULT_CODE:** 520
- **TYPE_CODE:** 4100
- **CODE:** DECLINE CHECK 5 UNPAID (ALL) AMT=$5478 GLOBAL eTELECOM 888-481-0757
- **MESSAGE:** DECLINE CHECK 5 UNPAID (ALL) AMT=$5478 GLOBAL eTELECOM 888-481-0757

If a transaction is declined the Authorization Gateway will return an 8 in the RESULT_CODE element which indicates a Decline Message. In the returned response above the RESULT_CODE element has a value of 520. If the host system is setup to do a bit comparison of the value in the RESULT_CODE you will discover that the 520 is made of an 8 indicating a Decline Message and 512 which indicates that the unpaid check Limit has been exceeded. The fixed decline response also contains a value of 4100 in the TYPE_CODE element. This again indicates an internal override was done because the Authorization Gateway is returning a predetermined fixed response. However, if the host system is setup to conduct a bit comparison on the value of the TYPE_CODE element it can also be determined that TYPE_CODE also contains a 4, which indicates a Business Check was sent for processing.

15.6 Process Single Certification Check – Void

When voiding a previously approved single certification check you will need to invoke the ProcessSingleCertificationCheck web method and set the routing number to 490000018, 490000021, or 490000047 in the ROUTING_NUMBER element of the request XML Data Packet. You will also have to set the value of the IDENTIFER element to “V”. This milestone has been built into the development phase so that you can incorporate this functionality into your host system. If the request XML Data Packet is valid then a Void identifier for previously approved transaction with the routing numbers noted above will trigger the Authorization Gateway to return a response with the following information to the host system:

- **RESPONSE_TYPE:** A
- **RESPONSE_TYPE_TEXT:** APPROVED
- **RESULT_CODE:** 0
- **TYPE_CODE:** 5120
- **CODE:** VOID ACCEPTED
- **MESSAGE:** VOID ACCEPTED

You should note that the returned information for a voided transaction contains a RESULT_CODE of 0. This indicates that the void was approved, but it also illustrates the importance of examining the information contained with the TYPE_CODE. If the host systems interface with the Authorization Gateway was only set to interpret the RESULT_CODE the full meaning of the overall response would be lost. In this case the TYPE_CODE returned in the response XML Data Packet contains 5120. A bit comparison of this value indicates that the value contains 1024 indicating a voided check, and 4096 indicating that there was an internal override due to a predetermined fixed response being returned.

15.7 Process Single Certification Check – Reversal

When reversing a previously approved single certification check you will need to invoke the ProcessSingleCertificationCheck web method and set the routing number to 490000018, 490000021, or 490000047 in the ROUTING_NUMBER element of the request XML Data Packet. You will also have to set the value of the IDENTIFER element to “F”. This milestone has been built into the development phase so that you can incorporate this functionality into your host system. If the request XML Data Packet is valid then a Reversal identifier for previously approved transaction with the routing numbers noted above will trigger the Authorization Gateway to return a response with the following information to the host system:

- **RESPONSE_TYPE:** A
- **RESPONSE_TYPE_TEXT:** APPROVED

- **RESULT_CODE:** 0
- **TYPE_CODE:** 5120
- **CODE:** REVERSAL ACCEPTED
- **MESSAGE:** REVERSAL ACCEPTED

15.8 Process Single Certification Check – Credit

When processing a single certification check for Authorization you will need to invoke the ProcessSingleCertificationCheck web method. A credit transaction is processed with a negative sign in front of the amount. Set the routing number to 490000018 in the ROUTING_NUMBER element of the request XML Data Packet. You will also have to set the value of the IDENTIFER element to “R” if you are using a PPD or CCD schema or “A” for all other schemas. If the request XML Data Packet is valid then this routing number will trigger the Authorization Gateway to return a response with the following information to the host system:

- **RESPONSE_TYPE:** A
- **RESPONSE_TYPE_TEXT:** APPROVED
- **RESULT_CODE:** 0
- **TYPE_CODE:** 4096
- **CODE:** AUTH NUM 272-172
- **MESSAGE:** APPROVAL

15.9 Process Single Certification Check – Manager Needed

When processing a single certification check for Manager Needed you will need to invoke the ProcessSingleCertificationCheck web method and set the routing number to 490000021 in the ROUTING_NUMBER element of the request XML Data Packet. You will also have to set the value of the IDENTIFER element to “R” if you are using a PPD or CCD schema or “A” for all other schemas. If the request XML Data Packet is valid then this routing number will trigger the Authorization Gateway to return a response with the following information to the host system:

- **RESPONSE_TYPE:** W
- **RESPONSE_TYPE_TEXT:** Warning
- **RESULT_CODE:** 132
- **TYPE_CODE:** 4100
- **CODE:** MANAGER NEEDED CHECK TOO LARGE
- **MESSAGE:** MANAGER NEEDED CHECK TOO LARGE

If a transaction is returned with a warning message the RESULT_CODE element will contain a 4. In the above response message a bit comparison will show that the RESULT_CODE element also contains a 128 which indicates that the transaction limit has been exceeded. The host system should be setup to recognize warning messages and any other additional information contained within the RESULT_CODE element. In this case the MESSAGE element indicates that the Manager is needed because the check is too large. The TYPE_CODE again shows that 4096 was returned indicating that there was an internal override due to a predetermined fixed response being returned as well as a 4 indicating a Business Check was sent for processing.

If the host system receives a warning message back and indicates “MANAGER NEEDED” and you are not doing PPD, then you have the option of sending an override request packet back to

the Authorization Gateway. The override request is created by sending the same request XML Data Packet back to the Authorization Gateway. However, you must change the value of the IDENTIFIER element to “O”. You also have the option of changing the values of the TRANSACTION_ID element and REQUEST_ID attribute so that your host system can record and track the override request as a separate transaction. In this instance you also have the option of changing the value in the CHECK_AMOUNT element. Again, identifying a request XML Data Packet as an override will void the previous transaction and input a new transaction in its place, and your host system will receive an authorization in return.

15.10 Process Single Certification Check – Represented Check

When processing a single certification check for Re-Presented Check you will need to invoke the ProcessSingleCertificationCheck web method and set the routing number to 490000047 in the ROUTING_NUMBER element of the request XML Data Packet. You will also have to set the value of the IDENTIFER element to “R” if you are using a PPD or CCD schema or “A” for all other schemas. If the request XML Data Packet is valid then this routing number will trigger the Authorization Gateway to return a response with the following information to the host system:

- **RESPONSE_TYPE:** W
- **RESPONSE_TYPE_TEXT:** Warning
- **RESULT_CODE:** 4
- **TYPE_CODE:** 4228
- **CODE:** MANAGER NEEDED REPRESENTED CHECK
- **MESSAGE:** MANAGER NEEDED REPRESENTED CHECK

If a transaction is returned with a warning message the RESULT_CODE element will contain a 4. In this case the MESSAGE element indicates that the Manager is needed because this is a represented check. The TYPE_CODE in this response contains a lot of information. A bit comparison will show that the value of 4228 in the TYPE_CODE element contains 128 which indicates a represented check as well as 4096 indicating that there was an internal override due to a predetermined fixed response being returned, and a 4 indicating a Business Check was sent for processing. The host system should be able to recognize that a warning message was received from the Authorization Gateway and that it was a represented check.

If the host system receives a warning message back and indicates “MANAGER NEEDED” and you are not doing PPD, then you have the option of sending an override request packet back to the Authorization Gateway. The override request is created by sending the same request XML Data Packet back to the Authorization Gateway. However, you must change the value of the IDENTIFIER element to “O”. You also have the option of changing the values of the TRANSACTION_ID element and REQUEST_ID attribute so that your host system can record and track the override request as a separate transaction. Again, identifying a request XML Data Packet as an override will void the previous transaction and input a new transaction in its place, and your host system will receive an authorization in return.

16.0 Exception Handling

Incorporating exception handling into your host system for the Authorization Gateway interface is important so that the host system can check to ensure that nothing unexpected occurred

during processing. There are basically two reasons the Authorization Gateway may return an exception in the response XML Data Packet.

Although the Authorization Gateway has been rigorously tested it is possible that an internal error may occur. If this happens the Authorization Gateway will return an EXCEPTION element with a message of “An internal error occurred. The Transaction was NOT processed”. If this exception is return to the host system our software team will be immediately notified with detailed information about the problem, and will work to correct the issue. We work hard to ensure these types of exceptions do not occur, but your integration team should understand what internal errors mean, how they are handled, and configure the host system to take appropriate action.

The Authorization Gateway may also return exception messages if there are authentication, authorization, or data related errors. The message for these types of exceptions will vary, but the host system will receive a detailed message within the EXCEPTION element that outlines exactly what the problem was. These types of exceptions are built into the Authorization Gateway by design and the Authorization Gateway relies on the host system to resolve the issue.

We expect that your integration team has included at least a minimal level of exception handling into your host system prior to beginning the Certification Phase.

If an error occurs within the Authorization Gateway the XML string response will detail the reason for the error within an Exception element. The Exception element will NOT be present if an error did not occur. However, should an error occur, the Exception element may be found as a child element of either the Response element, or the Transaction element.

16.1 EXCEPTION Element – Example as a child of the RESPONSE element

```
<? Xml version=”1.0” encoding=”utf-8” ?>
  <RESPONSE xmlns:xsd=”http://www.w3.org/2001/XMLSchema”
               xmlns:xsi=”http://www.w3.org/2001/XMLSchema-instance”
    REQUEST_ID=”” >
       <EXCEPTION>
      <MESSAGE>An internal error occurred. The transaction was NOT
               Processed.
             </MESSAGE>
    </EXCEPTION>
     </RESPONSE>
```

The Exception element will contain the following elements.

- MESSAGE: Contains text information about the exception.

17.0 Request an Archived Response

Each time a valid data packet request is processed the Authorization Message that is returned is archived. To maintain a high level of performance Authorization Messages are archived asynchronously while the original Authorization Message is returned to the requestor.

If needed an Authorization Message for a previously processed transaction can be requested again by invoking the GetArchivedResponse web method. It is important to note that the transaction is not processed again, only the original Authorization Message that was archived is returned. Each Authorization Message is archived along with the unique user defined Request ID and Terminal ID that was provided in the data packet request. The GetArchivedResponse web method accepts the Request ID as an input parameter and will return the original Authorization Message for the given Request ID and Terminal ID.

***NOTE**: If Authorization Gateway Request IDs are duplicated for a given Terminal, only the last Authorization Message for the pairing will be returned.*

18.0 Requesting a Certification Script

Requesting a certification script is the major milestone of the Development Phase. It signifies that your integration team has completed the integration effort and alerts our software team that the host system is ready to undergo certification. It is important that your integration team contact us to request a certification script. If a certification script is not requested, but you begin the Certification Phase, our software team will not be able to properly certify your host system and your team will have to rerun the certification script prior to moving to the Production Phase.

19.0 Beginning Certification (Phase 3)

During the certification phase your integration team will be responsible for sequentially completing the objectives in the certification script provided. Your team should now be intimately familiar with the Authorization Gateway and the host system should now be able to handle the completion of these objectives without any problems. During the certification phase our software team will closely monitor each transaction to ensure it is valid, and that the host system is properly configured. We will alert you to the status of the transaction in our system, and advise you if there are any modifications that need to be made to the host system. The successful completion of each objective outlined in the certification script signifies the completion of the major milestone for the Certification Phase and marks the opportunity to begin the Production Phase.

20.0 Migrating to Production (Phase 4)

The Production Phase is the final phase of the integration effort. During this phase you will have to make some minimal changes to the host system in order to use the Authorization Gateway in a production environment. This includes the following:

- You will need to request a user name and password for production. This user name and password will be different from the user name and password provided for certification

and will only be valid for production. The host system will then need to be modified to include this user name and password in the authentication header when invoking a production web method.

- You will need to request the production URL for the Authorization Gateway. This URL will be different then the URL used for certification, however it will contain identical web methods. The host system will need to be modified to invoke web methods on the production URL.
- You will also need to change the certification web methods listed below to their sister production web methods.

Certification Web Method	Production Web Method
GetCertificationTerminalSettings	GetTerminalSettings
ProcessSingleCertificationCheck	ProcessSingleCheck

- Once the host system has been modified to include these changes for processing transactions in a production environment you will need to request a "Go Live" date. Requesting a "Go Live" date signifies completion of the last major milestone of the integration effort and indicates to our software team that the host system is ready for production.

21.0 Authorization Requirements

21.1 Authorization Page – PPD/CCD

For prearranged payment and deposit (PPD/CCD) entries the receiver must have the following text listed on the authorization page of their site.

- "Submission of this transaction assumes an agreement is in place between both parties to allow converting this transaction into an Electronic Funds Transfer transaction or paper draft, and to debit this account for the amount of the transaction. Additionally, the agreement further states that in the event this draft or EFT is returned unpaid, a service fee, as allowable by law, will be charged to this account via draft or EFT."

21.2 Authorization Page – WEB

For internet initiated (WEB) entries the receiver must have the following text listed on the authorization page of their site.

- "By authorizing this transaction, customer agrees that merchant may convert this transaction into an Electronic Funds Transfer (EFT) transaction or paper draft, and to debit this account for the amount of the transaction. Additionally, in the event this draft or EFT is returned unpaid, a service fee, as allowable by law, will be charged to this account via EFT or draft."

Merchants are required to retain the original authorization or copy of the original authorization in its original form that can be reproduced upon request. NACHA does not accept proof of an authorization as being a listing of the information captured at time of authorization.
The following information must be included in the authorization record:

- Consumer IP Address of Origination
- Consumer Name
- Consumer Address
- Transaction Amount
- Transaction Effective Date
- Consumer E-mail address (optional; industry recommended best practice)
- Website where payment was accepted
- Signifying whether authorization is for a single or recurring/multiple debits, and debit schedule if recurring/multiple
- Consumer Banking information
- Statement of how the consumer's identity was authenticated

21.3 Recorded Authorization – TEL

For telephone initiated (TEL) entries the receiver must have the following verbiage (or substantially similar) read and captured on the recorded customer authorization.

- "By providing your bank account information and verbal authorization today, <Today>, you are authorizing <Merchant> to create an ACH debit to your account and that this Check by Phone may be drafted from your account as early as today. In the event your Check by Phone is returned from your bank unpaid, you further agree that a fee of $<Fee> or as allowable by law shall also be charged to your account via draft or ACH debit. Do you authorize <Merchant> to proceed with this Check by Phone? A Check by Phone will be drafted from your bank account with the following information(Bank Routing Number, Account Number, Check Number, and Check by Phone Amount). Please allow 12 to 72 business hours for this transaction to post to your account. Should you have any questions regarding your payment, you may reach our office at <Contact Phone>."

21.4 Receipt Authorization – POP

For point-of-purchase (POP) single debit entries the receiver must receive a copy of the receipt and the voided check. The receipt provided to the receiver must contain the following for each of the transaction types below.

- **Approved Sale or Override:**
  - **Footer Text:** "I authorize the merchant to convert my check to an electronic funds transfer, or paper draft, and debit my account for the amount of the transaction. In the event that my draft or EFT is returned unpaid, I agree that a fee of $25 (or as allowed by law) may be charged to my account via draft or EFT."
  - **Signature Line:** Yes
- **Verification Only:**
  - **Footer Text:** "Must retain check for deposit."
  - **Signature Line:** No
- **Decline:**
  - **Footer Text:** None
  - **Signature Line:** No
- **Void:**

- **Footer Text:** None
- **Signature Line:** No

22.0 Sample Code – Microsoft Visual Studio 2003

The first step is to add a Web Reference to the web service URL below in your project called com.eftchecks.demo.

https://demo.eftchecks.com/Webservices/AuthGateway.asmx

22.1 VB.NET

Example Code – GetCertificationTerminalSettings()

```
Public Function GetCertificationTerminalSettings() As String
    'This function will get the Certification Terminal Settings for Terminal 1010.

    'Create variable to hold Authorization Gateway Response
    Dim myAuthGatewayResponse As String

    'Create an instance of the Authorization Gateway
    Dim myAuthGateway As New com.eftchecks.demo.AuthGateway

    'Create an instance of the Authorization Header
    Dim myAuthHeader As New com.eftchecks.demo.AuthGatewayHeader

    'Populate the Auth Header with the User Name, Password, and Terminal ID
    With myAuthHeader
        .UserName = "myUserNameGoesHere"
        .Password = "myPasswordGoesHere"
        .TerminalID = 1010
    End With

    'Apply the Auth Header to the Auth Gateway
    myAuthGateway.AuthGatewayHeaderValue = myAuthHeader

    'Get the Certification Terminal Settings from the Authorization Gateway
    myAuthGatewayResponse = myAuthGateway.GetCertificationTerminalSettings()

    'Create a new XML Document for the Certification Terminal Settings
    Dim myTerminalSettings As New System.Xml.XmlDocument

    'Load the Certification Terminal Settings XML into an XML Document
  myTerminalSettings.LoadXml(myAuthGatewayResponse)

  'Return the Certification Terminal Settings
  Return myTerminalSettings.OuterXml.ToString

End Function
```

22.2 C#

```
public string GetCertificationTerminalSettings()
{
  //This function will get the Certification Terminal Settings for Terminal 1010.

  //Create variable to hold Authorization Gateway Response
  string myAuthGatewayResponse;

  //Create an instance of the Authorization Gateway
  com.eftchecks.demo.AuthGateway myAuthGateway = new
com.eftchecks.demo.AuthGateway();

  //Create an instance of the Authorization Header
  com.eftchecks.demo.AuthGatewayHeader myAuthHeader = new
  com.eftchecks.demo.AuthGatewayHeader();

  //Populate the Auth Header with the User Name, Password, and Terminal ID
        {
          myAuthHeader.UserName = "myUserNameGoesHere";
          myAuthHeader.Password = "myPasswordGoesHere";
          myAuthHeader.TerminalID = 1010;
        }

  //Apply the Auth Header to the Auth Gateway
  myAuthGateway.AuthGatewayHeaderValue = myAuthHeader;

  //Get the Certification Terminal Settings from the Authorization Gateway
  myAuthGatewayResponse = myAuthGateway.GetCertificationTerminalSettings();

  //Create a new XML Document for the Certification Terminal Settings
  System.Xml.XmlDocument myTerminalSettings = new System.Xml.XmlDocument();

  //Load the Certification Terminal Settings XML into an XML Document
  myTerminalSettings.LoadXml(myAuthGatewayResponse);

  //Return the Certification Terminal Settings
  return myTerminalSettings.OuterXml.ToString;

}
```

22.3 SOAP Message Sample

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
```

```
    <AuthGatewayHeader
xmlns=”http://tempuri.org/GETI.eMagnus.WebServices/AuthGateway” >
      <UserName> GATEWAYUserName </UserName>
      <Password> GATEWAYPassword</Password>
      <TerminalID>1210</TerminalID>
    </AuthGatewayHeader>
  </soap:Header>
  <soap:Body>
    <ProcessSingleCertificationCheck
xmlns=”http://tempuri.org/GETI.eMagnus.WebServices/AuthGateway”>
      <DataPacket>&lt;?xml version=&quot;1.0&quot; encoding=&quot;utf-8&quot;
standalone=&quot;no&quot;?&gt;
       &lt;AUTH_GATEWAY REQUEST_ID=
&quot;8949a6093fbc414b871eb65e019a8f08&quot;&gt;
       &lt;TRANSACTION&gt;
       &lt;TRANSACTION_ID&gt;&lt;/TRANSACTION_ID&gt;
       &lt;MERCHANT&gt;
       &lt;TERMINAL_ID&gt;1210&lt;/TERMINAL_ID&gt;
       &lt;/MERCHANT&gt;
       &lt;PACKET&gt;
       &lt;IDENTIFIER&gt;A&lt;/IDENTIFIER&gt;
       &lt;ACCOUNT&gt;
       &lt;ROUTING_NUMBER&gt;490000018&lt;/ROUTING_NUMBER&gt;
       &lt;ACCOUNT_NUMBER&gt;999999&lt;/ACCOUNT_NUMBER&gt;
       &lt;CHECK_NUMBER&gt;9999&lt;/CHECK_NUMBER&gt;
       &lt;ACCOUNT_TYPE&gt;Checking&lt;/ACCOUNT_TYPE&gt;
       &lt;/ACCOUNT&gt; &lt;CONSUMER&gt;
       &lt;FIRST_NAME&gt;Doug&lt;/FIRST_NAME&gt;
       &lt;LAST_NAME&gt;Fresh&lt;/LAST_NAME&gt;
       &lt;ADDRESS1&gt;22 West Way&lt;/ADDRESS1&gt;
       &lt;ADDRESS2&gt;&lt;/ADDRESS2&gt;
       &lt;CITY&gt;Los Angls Afb&lt;/CITY&gt;
       &lt;STATE&gt;CA&lt;/STATE&gt;
       &lt;ZIP&gt;90009&lt;/ZIP&gt;
       &lt;PHONE_NUMBER&gt;2073331234&lt;/PHONE_NUMBER&gt;
       &lt;DL_STATE&gt;&lt;/DL_STATE&gt;
       &lt;DL_NUMBER&gt;&lt;/DL_NUMBER&gt;
       &lt;COURTESY_CARD_ID&gt;&lt;/COURTESY_CARD_ID&gt;
       &lt;/CONSUMER&gt;
       &lt;CHECK&gt;
       &lt;CHECK_AMOUNT&gt;24.55&lt;/CHECK_AMOUNT&gt;
       &lt;/CHECK&gt;
       &lt;/PACKET&gt;
       &lt;/TRANSACTION&gt;
       &lt;/AUTH_GATEWAY&gt;
      </DataPacket>
    </ProcessSingleCertificationCheck>
  </soap:Body>
</soap:Envelope>
```

23.0 Contact Information

For questions or to receive certification and live username/passwords and URLs please contact:

Integration Department
integration@eftsupport.com

24.0 Document History

Version Number	Modification Date	Modification
1.1	03/21/2008	Updated MICR_DATA element description and data type
1.2	04/02/2008	Updated to include EXCEPTION element information
1.3	05/08/2008	Updated to include GetArchivedResponse Web Method
1.3	05/08/2008	Updated to include Line Number and Line Position attributes on the Validation Error Element
1.4	05/13/2008	Updated to include Check 21 Specification
1.5	05/15/2008	Updated to include requested Username and Password.
1.5	05/15/2008	Updated to include Process Single Certification Check Web Method.
1.6	07/31/2008	Update Courtesy Card Information
1.7	02/27/2009	Updated to include CCD specification & Credits for PPD and CCD
1.8	4/12/2010	Added Void and Reversal to all SEC Codes
1.9	6/16/2010	Added non-guarantee schemas to PPD, CCD TEL and WEB
1.10	9/28/2010	Updated non-guarantee & credit schema links to PPD, CCD TEL and WEB
2.0	12/13/2011	Added Authorization Verbiage for each SEC code, updated Contact Information and added Custom field descriptions
2.1	2/29/2012	Add CCD definition
2.2	3/13/2012	Removed WEB Schemas – Guaranteed and terminals from the documentation. Corrected misspelling issues.
2.3	4/17/2012	Corrected Ck21 hyperlinks, added Token & MICR information & removed ARC & BOC information
2.4	6/15/2012	Added additional Web authorization requirements
2.5	10/08/2012	Added IPVerification documentation
2.6	4/1/2014	Rebranded document
2.7	4/4/2014	Merged Implementation guide and specs into one document
2.8	6/20/2014	Updated terminal id reference to numeric value