



Conexión REST

Manual de Integración API Java

Versión: 1.0

Fecha: 01/06/2018

Referencia: RS.TE.CEL.MAN.0025



Redsys, Servicios de Procesamiento, S.L. – c/ Francisco Sancha, 12 – 28034 Madrid (España)

www.redsys.es

Autorizaciones y control de versión

AUTOR: Redsys	VALIDADO POR: Comercio Electrónico	APROBADO POR: Redsys
Empresa: Redsys	Empresa: Redsys	Empresa: Redsys
Firma:	Firma:	Firma:
Fecha:	Fecha: 01/06/2018	Fecha: 01/06/2018

Versión	Fecha	Afecta	Breve descripción del cambio
1.0	01/06/2020	TODO	Versión Inicial

ÍNDICE

1. OBJETIVO	4
2. ESTRUCTURA DE LA API	4
2.1 PAQUETE ES.REDSYS.REST.API.CONSTANS	4
2.2 PAQUETE ES.REDSYS.REST.API.MODEL	5
2.3 PAQUETE ES.REDSYS.REST.API.MODEL.ELEMENT	5
2.4 PAQUETE ES.REDSYS.REST.API.MODEL.MESSAGE	5
2.5 PAQUETE ES.REDSYS.REST.API.SERVICE	6
2.6 PAQUETE ES.REDSYS.REST.API.SERVICE.IMPL	6
2.7 PAQUETE ES.REDSYS.REST.API.UTILS	6
2.8 PAQUETE ES.REDSYS.REST.TEST.EXAMPLE	6
3. CREACIÓN DEL SERVICIO DE CONEXIÓN	7
4. PARÁMETROS	7
4.1 PARÁMETROS DE ENVÍO OBLIGATORIOS EN UNA PETICIÓN	8
4.2 PARÁMETROS DE ENVÍO OPCIONALES	9
4.3 PARÁMETROS Y FUNCIONES. PETICIÓN RESTOPERATIONSERVICE	10
4.4. PARÁMETROS Y FUNCIONES. PETICIÓN RESTINITIALREQUESTSERVICE	12
4.5. PARÁMETROS Y FUNCIONES. PETICIÓN RESTAUTHENTICATIONREQUESTSERVICE	13
4.6 PARÁMETROS DE RESPUESTA	13
5. EJEMPLOS	16
6. REQUISITOS Y ESPECIFICACIONES TÉCNICAS	17

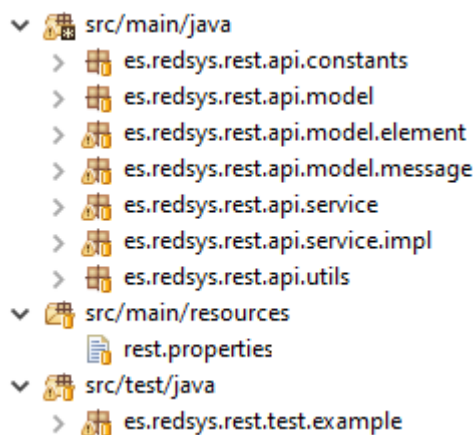
1. Objetivo

El presente documento define la guía de integración de la conexión REST para la integración de un comercio con el TPV-Virtual de Redsys mediante el uso de la API en lenguaje de programación Java.

2. Estructura de la API

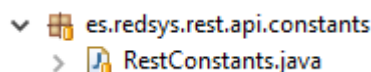
Para utilizar la API, el comercio debe instalar en su sistema el paquete que Redsys le ha proporcionado. En este caso con agregar al Build Path de su proyecto la librería es suficiente.

Una vez importada la librería, el comercio ya puede hacer uso de las clases y operativas necesarias para enviar los datos de la operación. La API consta los siguientes elementos:



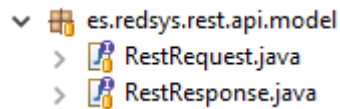
NOTA: dentro de src/main/resources se encuentra el fichero (rest.properties) con las propiedades de la API modificables (tiempo de respuesta, definición del servicio...). Se debe tener especial cuidado al manipularlo ya que una configuración errónea puede inutilizar la API completamente.

2.1 Paquete es.redsys.rest.api.constans



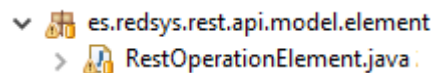
En este paquete encontramos el nombre y valor de las constantes utilizadas en el intercambio de mensajes (petición y respuesta) vía REST.

2.2 Paquete es.redsys.rest.api.model



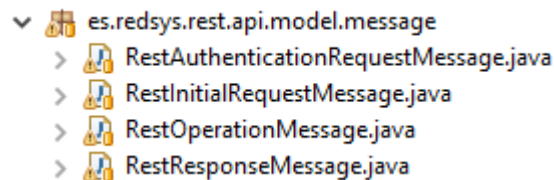
Este paquete contiene las dos interfaces de definición de tipos de mensajería utilizada en la API (petición y respuesta).

2.3 Paquete es.redsys.rest.api.model.element



Los elementos para capturar y crear la respuesta del mensaje API se encuentran en este archivo.

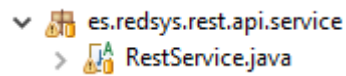
2.4 Paquete es.redsys.rest.api.model.message



Dentro de este paquete se encuentran las clases con los diferentes mensajes que se pueden tratar en la API y que se intercambiarán entre los sistemas del comercio y Redsys. Dentro de cada una de las clases se encuentran los parámetros y métodos que pueden ser usados para cada tipo de mensaje.

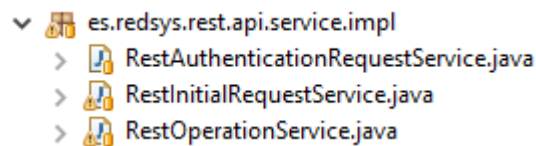
NOTA: ver apartado "4. Parámetros" para ver la lista de parámetros de petición y respuesta, y para obtener más información sobre los métodos a usar en cada una de las operaciones

2.5 Paquete es.redsys.rest.api.service



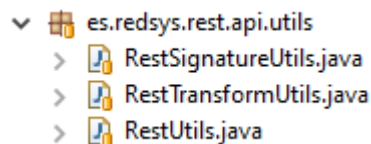
Este paquete contiene el contenedor de la clase general para la realización de la llamada REST.

2.6 Paquete es.redsys.rest.api.service.impl



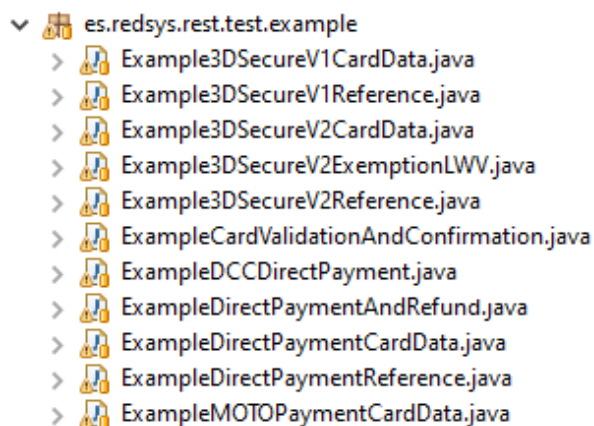
Aquí se encuentran las clases implementando los diferentes tipos de operaciones que se pueden realizar a través de la API.

2.7 Paquete es.redsys.rest.api.utils



Este paquete contiene los métodos necesarios para realizar la llamada vía REST.

2.8 Paquete es.redsys.rest.test.example



Por último, se incluye un paquete de test (src/main/test) con distintos ejemplos del uso de la API dependiendo del tipo de operación que queramos hacer.

3. Creación del servicio de conexión

Para la conexión con Redsys, se deberá utilizar la clase abstracta `RestService` y, dependiendo del tipo de operación a realizar, se instanciará una clase u otra del paquete *es.redsys.rest.api.service*:

- **RestInitialRequestService:** Utilizaremos esta clase para pedir información sobre la tarjeta, tanto para la autenticación 3D Secure, y exenciones, como para el uso de la operativa DCC. La respuesta de esta petición nos permitirá conocer los datos de envío para estas funcionalidades que deberán enviarse en la clase "RestOperationService".
- **RestOperationService:** Clase general para realizar todas las operaciones de pago generales, tanto para realizar una operación sin autenticación, como con ella. En el segundo caso, dependiendo de la respuesta (frictionless o challenge) obtendremos la respuesta final de la operación, o será necesario una nueva petición con la clase `RestAuthenticationRequestService`.
- **RestAuthenticationRequestService:** Una vez realizada la operación con la clase "RestOperationService", utilizaremos esta clase para realizar una autenticación por parte del titular de la tarjeta y obtener la respuesta final de la operación.

A la hora de hacer una petición se instanciará una de las clases anteriormente expuestas (dependiendo del tipo de operación que queramos realizar) con dos parámetros:

- **Clave de firma del comercio (signature):** Clave de firma para operaciones. Este valor se puede consultar en el módulo de administración de canales. El almacenamiento de este valor debe ser seguro ya que una vulnerabilidad en el sistema del comercio puede ocasionar que el valor se comprometa y generar operaciones fraudulentas.
- **Entorno (environment):** Entorno al que se enviará el mensaje, ya sea pruebas (SANDBOX) o producción (PRODUCTION). **NOTA: Se puede definir una dirección diferente en este parámetro (CUSTOM).**

4. Parámetros

A la hora de enviar un mensaje a través de la API, se deben establecer las propiedades de dicho mensaje. Aquí se encuentran reflejados la lista de parámetros obligatorios, que deberán enviarse en cualquier operación que quiera realizarse mediante la API:

4.1 Parámetros de envío obligatorios en una petición

Éstos parámetros deberán enviarse siempre para cualquier tipo de operativa:

- **Importe (amount):** Importe de la operación. Para un importe real de 78,95 el valor informado en el campo será 7895 sin separador decimal. Los decimales dependerán del código de la moneda utilizada.
- **Moneda (currency):** Código internacional según normativa ISO-4217. El código para la moneda euros es 978.
- **Código de comercio (merchant):** Código FUC del comercio a utilizar, consultable a través del módulo de administración de canales. Para pruebas podemos utilizar el valor 999008881.
- **Código de terminal (terminal):** Código numérico de terminal a utilizar, consultable a través del módulo de administración de canales. Para pruebas podemos utilizar el 1.
- **Número de pedido (order):** Alfanumérico que tendrá el mismo valor que el enviado en la primera fase.
- **Tipo de operación (transactionType):** Haciendo uso de la clase *es.redsys.rest.api.constants.RestConstants.TransactionType* estableceremos el tipo de operación a realizar pudiendo utilizar los siguientes valores:
 - **Autorización (AUTHORIZATION)**
 - **Devolución (REFUND)**
 - **Preautorización (PREAUTHORIZACION)**
 - **Confirmación de preautorización (CONFIRMATION)**
 - **Cancelación de preautorización (CANCELLATION)**
 - **Borrar una referencia (DELETE_REFERENCE)**
 - **Validación tarjeta (VALIDATION)**
 - **Confirmación de la validación (VALIDATION_CONFIRMATION)**

Un ejemplo para el uso de estos parámetros mediante las funciones de la API sería:

```
RestOperationMessage request = new RestOperationMessage();
request.setAmount("123");
request.setCurrency("978");
request.setMerchant("999008881");
request.setTerminal("20");
request.setOrder(orderID);
request.setCardNumber("4548810000000003");
request.setCardExpiryDate("3412");
request.setCvv2("123");
request.setTransactionType(TransactionType.AUTHORIZATION);
```


4.2 Parámetros de envío opcionales

Son funciones para añadir parámetros opcionales a cada una de las peticiones:

- **Datos de tarjeta:** envío de datos de tarjeta, fecha de caducidad y cvv2:

```
request.setCardNumber("4548810000000003");
request.setCardExpiryDate("3412");
request.setCvv2("123");
```

- **Crear referencia:** genera una referencia con la tarjeta para que ésta se pueda utilizar en pagos posteriores.

```
request.createReference();
```

- **Usar referencia:** Tras la obtención de la referencia, es posible su uso para seguir realizando operaciones futuras.

```
request.useReference("0197620cb6e8a4d74b5597ff9ed9b28671e52a37");
```

- **Operación COF:** Es posible realizar la primera operación COF enviando como parámetro el tipo de operación COF que quiera realizarse:

```
request.setCOFOperation(RestConstants.REQUEST_MERCHANT_COF_TYPE_INSTALLMENTS)
```

Tipo de COF	Valor DS_MERCHANT_COF_TYPE
Installments	I
Recurring	R
Reauthorization	H
Resubmission	E
Delayed	D
Incremental	M
No Show	N
Otras	C

Para sucesivas operaciones:

```
request.setCOFTxnid("2006031152000");
```

- **Usar idOper:** Tras la obtención de la idOper mediante el iframe de integración Insite, es posible usarlo del siguiente modo:

```
request.setOperID(operID);
```

- **Otros parámetros:** En el caso que quiera usarse un parámetro opcional que no esté entre los anteriores, podrá utilizarse la función "addParameter" enviándose en ella el nombre del parámetro con su valor de la siguiente forma:

```
request.addParameter("DS_MERCHANT_PRODUCTDESCRIPTION", "Valor del parámetro opcional");
```

- **DS_MERCHANT_EMV3DS:** En el caso en el que se quiera añadir algún parámetro a este elemento de la autenticación, podrá usarse la siguiente función para cada uno de los valores de la petición. Por ejemplo:

```
request.addEmvParameter("protocolVersion", "1.0.2");
```

NOTA: Para consultar el resto de parámetros opcionales, consultar la guía de parámetros.

4.3 Parámetros y funciones. Petición RestOperationService

Esta clase es para la realización de una operación, ya sea autenticando o mediante el pago directo:

- **Pago directo (direct payment):** Cuando se marca esta opción, la operación se realizará de forma no segura siempre que el comercio esté configurado para ello. No se solicitará autenticación al titular de la tarjeta.



```
request.useDirectPayment();
```

- **Pago MOTO:** La operación se realizará de forma no segura para las operativas tipo MOTO:

```
request.useMOTOPayment();
```

- **Uso de una exención:** Es posible usar una exención de autenticación:

```
request.setExemption("Valor de la exención a usar");
```

- **DCC:** Método para una operación de pago mediante la operativa DCC con los valores de la tarjeta de la petición inicial:

```
request.dccOperation(dccCurrency, dccAmount);
```

- **Parametros de autenticación con protocolVersion 1.0.2:** En el caso en el que se requiera una operación segura y el protocolVersion sea 1.0.2.

```
request.setEMV3DSParamsV1();
```

- **Parametros de autenticación con protocolVersion 2.X.0:** En el caso en el que se requiera una operación segura y el protocolVersion sea 2.X.0, se deberán enviar en la función los parámetros recogidos por el comercio de la operación 3DSMethod

NOTA: La operación 3DSMethod deberá ser realizada por el comercio. Para más información mirar el "Manual de Integración – REST. pdf"

```
request.setEMV3DSParamsV2(protocolVersion, browserAcceptHeader, browserUserAgent, browserJavaEnable,
    browserJavaScriptEnabled, browserLanguage, browserColorDepth, browserScreenHeight,
    browserScreenWidth, browserTZ, threeDSSTransID, notificationURL, threeDSCompInd);
```

NOTA: Estas dos últimas funcionalidades pueden no usarse a la hora de autenticar, y usar la función "addEmvParameter" para añadir uno a uno los parámetros de la petición de autenticación.

4.4. Parámetros y funciones. Petición RestInitialRequestService

Esta clase está creada para las operaciones basadas en un "iniciaPetición", es posible utilizar cualquiera de los parámetros anteriormente escritos (obligatorios y opcionales), a la vez que utilizar algunos específicos de ésta operativa:

- **Pedir información de la tarjeta:** Para ello existe una función específica en esta clase que puede usarse para recibir esta información en la petición.

```
request.demandCardData();
```

- **Tarjeta DCC:** Para consultar si se puede proceder a una operativa con DCC:

```
request.demandDCCinfo();
```

- **Pedir información de exenciones:** En el caso en el que se requiera la lista de exenciones que tiene disponible el comercio para la operativa de autenticación es posible utilizar la siguiente función:

```
request.demandExemptionInfo();
```

4.5. Parámetros y funciones. Petición RestAuthenticationRequestService

En el caso en el que se requiera una autenticación por parte del cliente (respuesta challenge), se deberá utilizar esta clase. Tras el tratamiento de los valores de autenticación que devuelve la petición "RestOperationService"(envío de los parámetros a autenticar), se deberán enviar los parámetros de vuelta de la url de autenticación a cada uno de estos métodos dependiendo del protocolVersion:

- **Ds_Merchant_Emv3ds versión 1.0.2:** En este caso, deberá enviarse los parámetros de respuesta a la vuelta de la autenticación del siguiente modo:

```
request.challengeRequestV1(pares, md);
```

- **Ds_Merchant_Emv3ds versión 2.X.0:** En este caso, tendrá que tenerse en cuenta la versión del protocolo, y deberá recogerse el valor cres para proceder a su envío:

```
request.challengeRequestV2(protocolVersion, cres);
```

4.6 Parámetros de respuesta

El servicio generará un mensaje de respuesta de tipo *RestResponse* que el comercio deberá analizar. En el paquete *es.redsys.rest.test.example* se incluyen ejemplos de varios tipos de operativa.

- **Resultado de la operación:** el análisis de la respuesta debe realizarse dependiendo del parámetro resultado (result) para cada una de las operativas.

```
response.getResult()
```

Existen tres posibilidades:

- a. La operación se ha tratado de forma correcta: el parámetro *result* será de tipo OK, el parámetro *apiCode* será 0 y el objeto *operation* vendrá informado con los datos de respuesta de la operación.

- b. Ha habido algún error en la operación: el parámetro *result* será de tipo KO, el parámetro *apiCode* tendrá un valor tipo SISXXXX y el objeto *operation* vendrá a nulo.
- c. La operación requiere autenticación: El parámetro *result* será de tipo AUT, el *apiCode* será 0 y en el objeto *operation* vendrá informado con los datos necesarios para la fase de autenticación.

Sí la operación ha resultado correctamente (valores "OK"/ "AUT"), pueden utilizarse las distintas funciones que la api pone a su disposición para analizar el mensaje.

- **Operaciones COF:** para obtener el id de la respuesta de las operaciones COF:

```
response.getCOFTxnid();
```

- **ThreeDSInfo:** Para obtener el tipo de respuesta de la autenticación:

```
String threeDSInfo = response.getThreeDSInfo();
```

- **ProtocolVersion:** En el caso en el que se realice un "iniciaPetición", en la respuesta estará el parámetro específico de la versión a utilizar para el siguiente paso de la operación. En el caso que se quiera autenticar, es necesario obtener el valor de este parámetro:

```
protocolVersion = response.protocolVersionAnalysis();
```

- **ThreeDSServerTransID y threeDSMethodURL:** En el caso que el protocolVersion sea del tipo "2.X.0" y se quiera autenticar, será necesario obtener los siguientes valores para realidad la petición de operación:

```
threeDSServerTransID = response.getThreeDSServerTransID();
```

```
threeDSMethodURL = response.getThreeDSMethodURL();
```

- **Exenciones:** En el caso en el que en la petición se pida la lista de exenciones que pueden ser usadas para el comercio, es posible obtener esta lista mediante la siguiente función:

```
response.getExemption();
```

- **DCC:** Para obtener la información de la tarjeta que aplique DCC en la respuesta de la petición inicial:

```
response.getDCCurrency();
response.getDCCAmount();
```

- **Obtención de parámetros para realizar la autenticación:** La respuesta de la operación puede ser de tipo "frictionless" o "challenge". En el caso en el primer caso se recibirá una respuesta OK, y en el segundo AUT. En el caso del challenge es necesario recoger los parámetros de la respuesta necesarios para enviarlos mediante un "POST" a la url de autenticación. Estos parámetros serán distintos dependiendo de la versión del protocolVersion :

```
//ProtocolVersion 1.0.2
String acsURL = response.getAcURLParameter();
String pAReq = response.getPAReqParameter();
String md = response.getMdParameter();
String termUrl = "http://www.comercio.com/authentication-response.jsp";
```

```
//ProtocolVersion 2.X.0
String acsURL = response.getAcURLParameter();
String cReq = response.getCReqParameter();
```

NOTA: Para obtener información sobre el tratamiento de estos datos y sobre el flujo de la operativa, ver guía la de "integración REST".

5. Ejemplos

Se pone a disposición una serie de ejemplos donde se muestran algunos casos de las operativas comunes y su forma de integración utilizando la API:

- **3DSecureV1CardData:** Ejemplo de un pago autorizado con datos de tarjeta y autenticación del titular versión 1.0.2
- **3DSecureV1Reference:** Ejemplo de un pago autorizado con referencia y autenticación del titular versión 1.0.2
- **3DSecureV2CardData:** Ejemplo de un pago autorizado con datos de tarjeta y autenticación del titular versión 2.X.0
- **3DSecureV2Reference:** Ejemplo de un pago autorizado con referencia y autenticación del titular versión 2.X.0
- **3DSecureV2ExemptionLWV:** Ejemplo de un pago autorizado con tarjeta donde se pide exención de autenticación.
- **CardValidationAndConfirmation:** Ejemplo de una validación de tarjeta, y posterior confirmación de la validación.
- **DCCDirectPayment:** Ejemplo de un pago autorizado sin autenticación y con tarjeta donde se hace uso de la operativa DCC.
- **DirectPaymentAndRefund:** Ejemplo de un pago autorizado con tarjeta y sin autenticación sobre el cual se hace posteriormente una devolución.
- **DirectPaymentCardData:** Ejemplo de un pago autorizado con tarjeta y sin autenticación.
- **DirectPaymentReference:** Ejemplo de un pago autorizado con referencia y sin autenticación.
- **MOTOPaymentCardData:** Ejemplo de un pago autorizado MOTO (sin autenticación).

NOTA: Para ejecutar los ejemplos es necesario añadir "-Dhttps.protocols=TLSv1.1,TLSv1.2" al argumento de la configuración JAVA a la hora de ejecutar el .java

NOTA: Los ejemplos proporcionados en el API son una representación de cómo utilizar las diferentes funciones del API y no se deben utilizar tal cual se proporcionan, ya que no contienen validaciones de seguridad y de negocio propias de la implementación de cada comercio. Redsys no

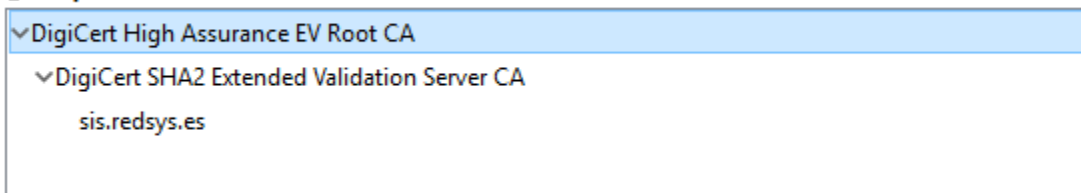
se hace responsable de la utilización de estos ejemplos en el servidor del comercio tal cual se proporcionan.

6. Requisitos y especificaciones técnicas

Los requisitos que debe cumplir el servidor de comercio para poder utilizar la API de conexión Java son los siguientes:

- Disponer de un servidor de aplicaciones que capaz de interpretar lenguaje Java en su versión 7 o superior.
- Habilitar la conexión de salida a las IP's de los servidores de Redsys, que son:
 - o 193.16.243.158
 - o 195.76.9.150
 - o 195.76.9.130
 - o 193.16.243.58
- Las CAs de los certificados de conexión son (se pueden obtener de <https://sis.redsys.es>):
 - o Autoridad de Certificación Raíz (DigiCert High Assurance EV Root CA)
 - o Autoridad de Certificación Intermedia (DigiCert SHA2 Extended Validation Server CA)

Jerarquía de certificados



- La lista de Cypher Suites compatibles es:
 - o TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
 - o TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384
 - o TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
 - o TLS_ECDH_RSA_WITH_AES_256_CBC_SHA
 - o TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
 - o TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256
 - o TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
 - o TLS_ECDH_RSA_WITH_AES_128_CBC_SHA
 - o TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384

- o TLS_ECDH_RSA_WITH_AES_256_GCM_SHA384
- o TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- o TLS_ECDH_RSA_WITH_AES_128_GCM_SHA256