WebService de reservas aéreo

Grupo Iris

Histórico de revisiones

Fecha	Usuario	Descripción
06/05/2016	Rubén	Se añade la posibilidad de recuperar una reserva a partir del localizador
09/12/2015	Coro	Se añade funcionalidad para permitir descuento de residentes (a partir de la versión 010)
17/08/2015	Rubén	Se añade la operación saveExternalPNR a partir de la version 009. Se añade usuario genérico de acceso a TEST2.
25/06/2014	Coro	Se añade información del SFPD.
09/11/2011	David	Se sustituyen todas las referencias a la versión 001 por la 002
21/06/2011	David	Se incluye documentación del proceso de emisión
18/05/2011	David	Creación del documento

AVISO MUY IMPORTANTE

Queda terminantemente prohibido la creación de cualquier tipo de demonio que realice ningún tipo de actividad recurrente.

La detección de cualquier tipo de proceso batch por parte de Grupo Iris conllevará la desconexión INMEDIATA del sistema.

Índice de contenido

1 Introducción

El objeto del presente documento es describir cómo se debe realizar el proceso completo de reserva de avión utilizando el WebService desarrollado por GrupoIris.

Este documento está orientado al personal técnico que va a realizar una integración utilizando el sistema de alguno de nuestros clientes.

2 Versiones

Cada vez que sea necesario incluir cambios en el WebService que conlleve cambios en la lógica del cliente o que necesite un cambio en el WSDL, se creará una nueva versión, con una nueva URL de acceso.

Las versiones anteriores se mantienen durante un periodo máximo de 6 meses. A partir de este momento, se eliminará la versión, por lo que tendrá que actualizar su sistema antes de este tiempo para que la aplicación siga activa.

3 Tecnología empleada

Los servicios están desarrollados utilizando WebServices con sesión, para lo cual se está utilizando el estándar WS-Addressing (http://www.w3.org/Submission/ws-addressing/). Tanto Java como .Net soportan este estándar.

4 Direcciones de acceso

4.1 Documentación en línea

La información detallada del WS se encuentra disponible en línea en http://javadocs.desarrollo.grupoiris.net/air/. Cuando acceda aquí, se le pedirá usuario y clave, debe introducir: javadoc/javadoc.

En esta dirección verá disponibles varias carpetas, de las cuales:

- Carpetas vXXX: es la documentación relativa a la versión XXX del WS.
- current: Siempre apunta a la última versión disponible.

Todos los enlaces incluidos en este documento, hacen siempre referencia a la última versión disponible.

4.2 Servidores

Existen tres direcciones distintas de acceso, dos de test y una de producción (que es única por cliente).

A todas las direcciones aquí especificadas hay que añadir la ruta que viene indicada en la tabla mostrada en Versiones disponibles, por ejemplo, la dirección del test para la versión 002 del WSDL de login es: http://test.aereo.grupoiris.net/**ws002/LoginManager?wsdl**.

Los desarrollos se realizarán siempre contra los sistemas de test y sólo una vez finalizado el desarrollo, y tras pasar una revisión por parte de GrupoIris, se permitirá el acceso al sistema de producción.

Los servidores de test están instalados en máquinas de pruebas que son más lentas que las de producción, los tiempos de respuesta son mucho mayores.

4.2.1 Accesos a TEST1 (Amadeus TEST)

Las peticiones de disponibilidad y las peticiones realizadas hacia Amadeus tienen un coste para el cliente que ha contratado el sistema aéreo.

Además, si se emite un billete y no se cancela inmediatamente o, en algunos casos, si se hace una reserva y no se cancela, es posible que esto también tenga costes para el cliente, llegando al 100% del precio en caso de billetes.

Para evitar tanto los costes por transacciones como los costes de los billetes o reservas, las primeras fases de desarrollo se realizan en un sistema de test en el que se utiliza el sistema de TEST de Amadeus.

El sistema de test de Amadeus no es 100% estable, por lo que habrá peticiones o momentos en el que el sistema no funcione adecuadamente. GrupoIris no puede hacer nada para solucionar estos problemas.

En este entorno no es necesario cancelar las reservas realizadas ya que no son reservas reales.

DIRECCIONES Y USUARIO DE ACCESO

- Servidor: http://test.aereo.grupoiris.net/
- Usuario de acceso: amadeus/amadeus.

4.2.2 Accesos a TEST2 (Amadeus en producción)

Una vez que los desarrolladores hayan realizado la primera fase de integración, y una vez que la agencia de viajes ya dispone de su propio acceso a Amadeus, es posible acceder a este

entorno de desarrollo.

La agencia de viajes tiene que autorizar esta configuración y sin esta autorización Grupo Iris no realizará las configuraciones oportunas para facilitar el acceso.

En este entorno de desarrollo el acceso a Amadeus es contra su sistema de producción utilizando el acceso de la agencia de viajes y con reservas reales. El acceso a lowcost es real, pero las reservas son ficticias ya que las compañías low cost no permiten la cancelación de las reservas.

En este entorno hay que tener en cuenta los siguientes detalles:

- Amadeus cobra a la agencia de viajes por las transacciones realizadas (es muy poco), NO REALIZAR NUNCA PRUEBAS DE CARGA.
- Amadeus cobra a la agencia de viajes por las peticiones de disponibilidad realizadas.
- Las reservas **son reales** y las compañías aéreas pueden cobrar a la agencia de viajes en determinadas ocasiones. En España sólo en algunos casos cobran si se dejan las reservas sin cancelar, pero en otros países nos han llegado incluso a pedir que no realicemos pruebas con nombres que incluyan "TEST" ya que pueden tener penalizaciones.

DIRECCIONES Y USUARIOS DE ACCESO

- Servidor: http://test2.aereo.grupoiris.net/
- Usuario de acceso: amadeus/amadeus.

El usuario anterior es un usuario genérico.

Los usuarios de acceso se crean para cada cliente, por lo que se les hará llegar un usuario propio cuando la agencia de viajes nos de autorización para activar este acceso.

4.2.3 Acceso a producción

Una vez realizado el desarrollo, y tras una revisión por parte de Grupo Iris, se le facilitará el acceso a la aplicación de producción.

Cada uno de nuestros clientes tiene una dirección de acceso distinto, por lo que le será facilitada cuando se le de acceso a dicha aplicación.

En este entorno todas las llamadas con todos los proveedores son 100% reales y serán facturadas a la agencia de viajes.

5 WebServices

5.1 LoginManager

WSDL: http://SERVIDOR/wsVERSION/LoginManager?wsdl por ejemplo, la versión 002 en la aplicación de test1, está disponible en: http://test.aereo.grupoiris.net/ws002/LoginManager?wsdl

Información detallada sobre este Servicio:
http://javadocs.desarrollo.grupoiris.net/air/current/com/gi/ws/package-summary.html

Operaciones disponibles:

5.1.1 login

5.1.1.1 Descripción

Realiza la operación de login y ofrece el endpoint sobre el que se realizarán las operaciones necesarias para el proceso de reserva.

5.1.1.2 Parámetros de entrada

- login: El login que se le haya facilitado para acceder al sistema
- password: La password

5.1.1.3 Respuesta

Devuelve el endpoint del servicio de reserva (un W3CEndpointReference).

5.1.1.4 Posibles errores

- InvalidLoginException: Si el login o el password especificados no es válido.

5.1.1.5 Más información

- Del servicio:
  http://javadocs.desarrollo.grupoiris.net/air/current/com/gi/ws/LoginManager.html#login%28java.lang.String,%20java.lang.String%29

5.2 Air

WSDL:http://SERVIDOR/wsVERSION/Air?wsdl, por ejemplo, en la versión 002 de la aplicación de test, está disponible en http://test.aereo.grupoiris.net/ws002/Air?wsdl.

- Información detallada del servicio:
  http://javadocs.desarrollo.grupoiris.net/air/current/com/gi/ws/air/package-summary.html
- Información detallada de las operaciones disponibles:
  http://javadocs.desarrollo.grupoiris.net/air/current/com/gi/ws/air/Air.html

5.2.1 getAutocompletePoint

5.2.1.1 Descripción

Realiza la búsqueda de los códigos IATA de las ciudades o aeropuertos a partir de la cadena de texto introducida por el usuario.

La operación implementa la lógica estándar de GrupoIris para mostrar el autocompletado que se le muestra al usuario al introducir una cadena de texto en el campo de origen o de destino.

Sólo busca a partir de 3 letras.

5.2.1.2 Entrada

`GetAutocompletePointRequest`:

- `point: Cadena de texto introducida por el usuario`

5.2.1.3 Salida

GetAutocompletePointReply

- pointList: List<AutocompletePoint>. Lista ordenada según las prioridades configuradas en el sistema de las opciones que deben ser mostradas al usuario. Cada una de ellas:
  - iataCode: El código IATA que debe ser enviado hacia la petición de disponibilidad.
  - description: Texto descriptivo que debe ser mostrado al usuario.

Devuelve una lista vacía en los siguientes casos:

- No se encuentra ninguna coincidencia con la cadena introducida.
- La cadena de texto introducida en la búsqueda es inferior a 3 letras.

5.2.2 GetMinBookingDate

5.2.2.1 Descripción

Permite obtener la fecha mínima de reserva. Cualquier disponibilidad con una fecha anterior a esta fecha no es resuelta por el sistema

5.2.2.2 Salida

La fecha es devuelta en el objeto GetMinBookingDateReply que sólo tiene un elemento "date".

5.2.3 GetCountries

5.2.3.1 Descripción

Permite obtener la lista de países. La lista de países es necesaria para los datos de contacto en caso de reservas de low cost y para los datos de nacionalidad y país de expedición de los documentos en caso de que sea necesario solicitar la información extra que exigen algunos países.

5.2.3.2 Salida

El objeto que contiene los países disponibles es: GetCountriesReply. Puede consultar la información completa de este objeto en:

http://javadocs.desarrollo.grupoiris.net/air/current/com/gi/ws/air/model/GetCountriesReply.html

El objeto contiene una lista de elementos de tipo CountryInfo, en el que se indica el nombre del país y su código IATA.

5.2.4 standardSearch

5.2.4.1 Descripción

Realiza la búsqueda de disponibilidad para vuelos de sólo ida o de ida y vuelta a partir de un origen y un destino.

El sistema realiza la búsqueda estándar definida en nuestro motor de reservas, obteniendo tarifas de GDS, de LowCost o incluso combinando, si así lo configura el cliente, opciones de sólo ida para realizar la búsqueda de disponibilidad.

La respuesta de la disponibilidad se mantiene en sesión, por lo que el sistema elimina cualquier búsqueda anterior de la sesión y la sustituye por esta.

5.2.4.2 Entrada

Recibe como parámetro un objeto de tipo StandardSearchRequest en el que se especifican los datos que el usuario a seleccionado para iniciar la búsqueda de disponibilidad:

- classOfService: uno de los valores definidos en http://javadocs.desarrollo.grupoiris.net/air/current/com/gi/ws/air/model/ClassOfService.html. La clase FIRST (primera) no está prácticamente disponible en ningún vuelo, por lo que no aconsejamos permitir esta búsqueda.
- departureDate. De tipo XMLGregorianCalendar (fecha y hora). Fecha deseada para el vuelo de ida. En caso de especificar hora distinta de las 0:00, la búsqueda se realiza para vuelos que salgan a partir de la hora especificada.
- DestinationIataCode: 3 letras. Es el código IATA de la ciudad o aeropuerto de destino. Uno de los valores devueltos por getAutocompletePoint en el campo iataCode.
- OnlyDirectFlights: Booleano. Por defecto FALSE. Si se establece a TRUE, el sistema sólo devolverá vuelos directos. Recomendamos que por defecto este valor esté a FALSE y que sea el usuario quien lo active en caso de querer sólo vuelos directos. En caso contrario el usuario final verá que no hay disponibilidad en su búsqueda si no hay ningún vuelo directo disponible y creerá que el sistema no está funcionando correctamente.
- OriginIataCode: 3 letras. Código IATA de la ciudad o aeropuerto de origen. Uno de los valores devueltos por getAutocompletePoint en el campo iataCode.
- PaxNumber: De tipo PaxNumber. Aquí se indica el número de pasajeros solicitados:
  - adult: Número de adultos para los que se realiza la búsqueda.
  - AdultResident: Número de adultos residentes. Sólo aplicable para vuelos entre la península y Canarias o Baleares o viceversa. Si se quiere pedir un vuelo para un residente, hay que enviar adultResident=1 y adult=0.
  - child: Número de niños para los que se realiza la búsqueda. Las compañías aéreas consideran niños a los que tienen entre 2 y 11 años (ambos incluidos) y no pueden viajar sin un adulto.
  - childResident: Número de niños residentes. Sólo aplicable para vuelos entre la península y Canarias o Baleares o viceversa. Si se quiere pedir un vuelo para un residente, hay que enviar childResident=1 y child=0.
  - infant: Número de bebés para los que se realiza la búsqueda. Las compañías aéreas consideran bebés a los menores de dos años. Los bebés no ocupan asiento y por lo tanto, sólo puede viajar un bebé por adulto.
  - infantResident: Número de bebés residentes.
- ReturnDate: De tipo XMLGregorianCalendar (fecha y hora). Fecha deseada para el vuelo de vuelta. En caso de especificar hora distinta de las 0:00, la búsqueda se realiza para vuelos que salgan a partir de la hora especificada. **Si no se especifica fecha de vuelta, el sistema realizará una búsqueda para un trayecto de sólo ida**

5.2.4.3 Salida

SearchReply: Consultar http://javadocs.desarrollo.grupoiris.net/air/current/com/gi/ws/air/model/SearchReply.html para ver la documentación detallada de los valores devueltos por la operación.

La operación devuelve una lista de propuestas, cada una de ellas contiene información sobre los siguientes aspectos:

- El precio, en el que se incluye tanto el precio total (desglosado en el precio de la tarifa,

las tasas y el fee a cobrar) como el precio por tipo de pasajero (también desglosado, sólo de los tipos de pasajero solicitados).

- Las notas de la tarifa.
- Las propuestas de vuelo:Las propuestas de vuelo están divididas en uno o varios JourneyGroups que a su vez contienen uno o varios Journeys, que a su vez contienen uno o varios Segment. A continuación se detalla su significado:
  - Hay un JourneyGroup cada trayecto solicitado, es decir, si el usuario ha seleccionado sólo ida esta lista contiene un sólo elemento con los diferentes formas de efectuar el trayecto de ida. Si el usuario ha seleccionado ida y vuelta, el primer elemento de la lista contiene las diferentes formas de efectuar el trayecto de ida y el segundo elemento las diferentes formas de efectuar la vuelta.
  - Cada JourneyGroup contiene uno o varios Journeys. Un Journey es un trayecto completo, es decir, contendrá la información para llegar desde el origen al destino, incluyendo los vuelos (Segment) que sean necesarios. Así, si un usuario solicita un MAD-NYC, un Journey puede contener un único Segment (MAD-NYC en el vuelo X de la compañía Y) y otro puede contener dos (un vuelo MAD-LON y otro LON-NYC).
  - El segment es cada uno de los vuelos que forman el Journey, incluye el número de vuelo (el que anuncian en los aeropuertos).

5.2.4.4 Posibles errores

- `InputErrorException` - Si hay algún error en la request enviada.
- `OperationErrorException` - Si se produce algún error en la ejecución de la búsqueda de disponibilidad

5.2.5 multiSearch

5.2.5.1 Descripción

Realiza la búsqueda de disponibilidad para múltiples itinerarios.

No se permite la búsqueda de residentes con múltiples itinerarios, las tarifas de residentes sólo son válidas para vuelos de ida o ida y vuelta entre las islas o entre las islas y la península.

La búsqueda se realiza únicamente en vuelos regulares y por lo tanto, no se incluyen compañías low cost.

Se puede pedir un máximo de tres trayectos.

La respuesta de la disponibilidad se mantiene en sesión, por lo que el sistema elimina cualquier búsqueda anterior de la sesión y la sustituye por esta.

5.2.5.2 Entrada

El objeto en el que se envía la información es: MultiSearchRequest. Puede ver la información detallada en http://javadocs.desarrollo.grupoiris.net/air/current/com/gi/ws/air/model/MultiSearchRequest.html

Debe de introducir los siguientes datos:

- classOfService: el tipo de tarifa a buscar (ver http://javadocs.desarrollo.grupoiris.net/air/current/com/gi/ws/air/model/ClassOfService.html).
- Journey: Lista de trayectos sobre la que se va a realizar la búsqueda de tipo

MultiSearchJourneyRequest. Como mínimo tiene que tener un elemento y como máximo 3. Cada uno de ellos con los siguientes datos:

- departureDate: la fecha del trayecto.
- OriginIataCode: el código IATA de la ciudad o aeropuerto de origen
- DestinationCode: el código IATA de la ciudad o aeropuerto de destino.

- OnlyDirectFlights: si se activa, sólo realizará búsquedas de vuelos directos.
- PaxNumber: El número de pasajeros.

5.2.5.3 Salida

La salida de este método es la misma que en el caso de standardSearch, excepto que la lista de JourneyGroup contiene un elemento por cada Journey que se haya introducido en la búsqueda.

5.2.5.4 Posibles errores

- `InputErrorException` - Si hay algún error en la request enviada.
- `OperationErrorException` - Si se produce algún error en la ejecución de la búsqueda de disponibilidad

5.2.6 quote

5.2.6.1 Descripción

Realiza la cotización de la propuesta y los trayectos elegidos para asegurar la tarifa y las plazas, así como obtener las notas completas de la tarifa.

Realiza las siguientes comprobaciones:

- Verifica la disponibilidad anterior
- Comprueba que el precio obtenido en disponibilidad es válido. Aunque es raro que suceda, es posible que existan pequeñas variaciones de precio entre la disponibilidad y la cotización
- Obtiene las notas completas de la tarifa.
- Permite saber qué datos hay que solicitar al usuario para finalizar la reserva

Esta operación debe ser efectuada una vez lanzada una operación de disponibilidad y sobre la misma sesión

A la hora de realizar la llamada es necesario enviar la referencia de la propuesta, así como las referencias de los trayectos elegidos en todos los tramos.

5.2.6.2 Entrada

El objeto que contiene los datos a enviar es: QuoteRequest, puede ver la documentación completa en la dirección:

http://javadocs.desarrollo.grupoiris.net/air/current/com/gi/ws/air/model/QuoteRequest.html.

En este objeto es necesario incluir las referencias tanto de la propuesta como de los journeys elegidos.

Ejemplo: El usuario selecciona la propuesta con referencia 1, el primer trayecto de ida (reference 0) y el segundo de vuelta (refernce 1). La petición sería:

```
proposalReference=1
journeyGroupReference= {
    journeyGroupReference=0, journeyReference=0
```

```
        journeyGroupReference=1, journeyReference=1
    }
```

5.2.6.3 Salida

El objeto que devuelve es QuoteReply, puede ver la documentación completa en:

http://javadocs.desarrollo.grupoiris.net/air/current/com/gi/ws/air/model/QuoteReply.html

Incluye la siguiente información:

- La propuesta actualizada
- Lista de avisos (warning) que deberían ser mostrados al usuario
- Información con los datos extra que deben ser solicitados al usuario
- Lista de ciudades necesarias para reservas de residentes
- Lista de comunidades autónomas para reservas de familia numerosa.

Importante: si la respuesta contiene el indicador checkPriceRequired, antes de realizar la llamada a la reserva (book), y habitualmente antes de realizar la llamada a la TPV, debe realizar una llamada a checkPrice.

Cuando el 100% o una parte de la reserva es de una compañía low cost, estas cobran el importe de la tarifa y las tasas directamente al usuario en su tarjeta de crédito, dentro de Proposal/QuoteInfo el campo noChargedPrice indica el importe que no será cobrado automáticamente en el proceso de reserva y que por lo tanto debe ser cobrado al usuario de otro modo.

5.2.6.4 Posibles errores

`InputErrorException` - Si hay algún error en the request enviada
`OperationErrorException` - Si se produce algún error en la cotización de la propuesta

5.2.7 checkPrice

5.2.7.1 Descripción

Esta operación es necesario llamarla siempre y cuando la operación de cotización (quote), devuelva el indicador checkPriceRequired a true.

Las compañías low cost incluyen gastos extra por pagar con determinadas tarjetas de crédito, por llevar maletas, etc. y ésta operación permite comprobar el precio final de la reserva antes de finalizar la reserva.

Esta operación debe realizarse antes de pasar por el proceso de la TPV.

5.2.7.2 Entrada

Los datos a enviar son los mismos que los de la operación de reserva (Book).

5.2.7.3 Salida

La respuesta de esta llamada es el objeto CheckPriceReply. La documentación completa de este objeto está accesible en:
http://javadocs.desarrollo.grupoiris.net/air/current/com/gi/ws/air/model/CheckPriceReply.html

Incluye los siguientes datos:

- La propuesta actualizada con los nuevos precios
- El indicador que permite saber si hubo o no cambio de precio en la comprobación.
- En Proposal/QuoteInfo, el campo noChargedPrice indica el valor que no se cobrará en la reserva (en la reserva sólo se cobra la tarifa y las tasas de las compañías de low cost).

5.2.7.4 Posibles errores

InputErrorException - Si hay algún error en la request enviada
OperationErrorException - Si se produce algún error en la cotización de la propuesta

5.2.8 Book

5.2.8.1 Descripción

Realiza la reserva de la propuesta elegida en la operación quote con los datos de los pasajeros que se indican el el request.

En caso de realizar más de una petición de quote para una misma disponibilidad, el sistema reserva la última llamada.

Muy importante: la respuesta de la operación quote(com.gi.ws.air.model.QuoteRequest), el elemento BookingRequiredData contiene la información sobre qué campos son obligatorios aún cuando el WSDL no los está especificando, bien por que se está realizando una reserva de LowCost, bien por que sea necesaria identificación extra,....

Si la respuesta incluye la variable lowcostPaymentUrl con una URL, es porque se está realizando una reserva de low cost y la compañía está pidiendo ir hacia el proceso de verified by visa del banco emisor de la tarjeta que ha introducido.
En estos casos, debe enviar el navegador del cliente hacia la URL que le llega en esta variable. Una vez finalizado el proceso de verificación, el sistema le confirmará la finalización del pago en la URL que usted haya especificado en BookRequest.lowcostPaymentReturnUrl, momento en el cual debe esperar la confirmación de la reserva llamando a la operación getBooking.

5.2.8.2 Entrada

BookRequest. Puede encontrar la documentación completa en http://javadocs.desarrollo.grupoiris.net/air/current/com/gi/ws/air/model/BookRequest.html

Los datos que deben ser enviados son los siguientes:

- Datos de contacto: obligatorio.
  - Email. Opcional.
  - Datos de contacto de lowcost: Datos de contacto solicitados por las compañías de lowcost. Obligatorio si BookingRequiredData.lowcostContactRequired es true.
- Tarjeta de crédito para el pago de reservas low cost: Obligatorio si BookingRequiredData.lowcostCreditCardRequired es true.
- Url a la que retornara la pasarela de pago de la compañía de lowcost. Obligatorio si BookingRequiredData.lowcostCreditCardRequired es true. Las compañías low cost realizan el cobro del importe de la reserva directamente en la tarjeta de crédito del viajero y actualmente todas ellas requieren el paso por el proceso de Verified By Visa o similar. Cuando este proceso finaliza, envían el navegador del cliente hacia la URL especificada aquí donde su aplicación debe llamar a la operación getBooking para verificar si la reserva se ha realizado correctamente.
- Datos de los pasajeros. Obligatorio un PaxInfo por cada pasajero solicitado:
  - Tipo de pasajero (adulto, niño, bebé, adulto residente, ...)
  - nombre. Obligatorio.
  - Apellidos. Obligatorio.
  - Fecha de nacimiento. Obligatorio si PaxRequiredData.birthDateRequired de la

respuesta de la cotización es true.

- Tratamiento del pasajero. Obligatorio. No inicie el campo con ningún valor en pantalla y oblige al usuario a seleccionar el valor adecuado, si se indica de forma incorrecta, el viajero puede tener problemas con la seguridad de los aeropuertos de llegada.
- Identificación del pasajero. Obligatorio si `PaxRequiredData.identificationRequired` de la respuesta de la cotización es true
- Código IATA (dos caracteres) del país de nacionalidad del pasajero. Obligatorio si `BookingRequiredData.securityFlightPassengerDataRequired` es true. La lista de todos los países se puede obtener con la operación GetCountries.
- Número de teléfono del pasajero. Obligatorio en el primer pasajero.
- Certificado de residencia. Obligatorio para pasajeros residentes, es decir si `PaxRequiredData.residentCredentialRequired` es true.

5.2.8.3 Salida

BookReply. Puede consultar la documentación completa en:
http://javadocs.desarrollo.grupoiris.net/air/current/com/gi/ws/air/model/BookReply.html

incluye todo aquello que es necesario para mostrar la reserva finalizada al usuario:

- Datos completos de la reserva
- En caso de ser una reserva con low cost, el sistema devuelve la url donde se debe dirigir el navegador del cliente para realizar el proceso de verified by visa

Reservas low cost

Si la respuesta incluye la variable lowcostPaymentUrl con una URL, es porque se está realizando una reserva de low cost y la compañía está pidiendo ir hacia el proceso de verified by visa del banco emisor de la tarjeta que ha introducido.
En estos casos, debe enviar el navegador del cliente hacia la URL que le llega en esta variable. Una vez finalizado el proceso de verificación, el sistema le confirmará la finalización del pago en la URL que usted haya especificado en `BookRequest.lowcostPaymentReturnUrl`, momento en el cual debe esperar la confirmación de la reserva llamando a la operación `Air.getBooking(com.gi.ws.air.model.GetBookingRequest)`.

El estado de la reserva para saber si la reserva ya se ha completado en la compañía low cost, está accesible en `BookingInfo.lowcostStatus`. El valor `LowcostStatus.UNCONFIRMED` indica que la reserva está pendiente de confirmar por la compañía aérea.

Tarjetas de pasajero frecuente

Desde la versión 003 de los WS es posible enviar las tarjetas de pasajero frecuente. Este dato va en la información de los pasajeros. SÓLO SE ENVÍAN TARJETAS DE PASAJERO FRECUENTE HACIA LAS COMPAÑÍAS REGULARES (las reservas hechas por GDS).

Las tarjetas de pasajero frecuente no tienen ningún estándar, por lo que no es posible realizar ningún tipo de validación previa. La tarjeta se envía hacia el GDS tal y como se recibe y si el GDS detecta algún tipo de error en dicha tarjeta, se aborta el proceso de reserva.

5.2.8.4 Posibles errores.

- `InputErrorException` - Si hay algún error en la request enviada

- `OperationErrorException` - Si se produce algún error en la reserva
- `PriceChangedException` - Si se detecta una variación en el precio en el momento de realizar la reserva. En cuyo caso, la "fault" del web service incluye la Proposal con el nuevo precio.

5.2.9 getBooking

5.2.9.1 Descripción

Permite recuperar una reserva por su identificador.

Debe utilizar esta operación al regresar del proceso de verified by visa para comprobar el estado de la reserva.

Las reservas low cost habitualmente se confirman en el momento, pero es posible que queden en un estado UNCONFIRMED durante varias horas. Su implementación debe mantener a la espera al usuario durante el tiempo máximo que estime oportuno (en torno a un minuto) y si la reserva no se confirma, mostrarle un aviso que le indique que la reserva puede estar realizada y por lo tanto no será posible devolverle el dinero. Durante este tiempo que mantiene al usuario en espera, debe estar realizando llamadas a getBooking cada 10 segundos.

Nuestro sistema mantiene un demonio que está revisando estar reservas cada 5 minutos y cuando se confirma o da por fallida, el sistema envía un correo a la dirección de contacto especificada en la petición de la reserva.

Si desea realizar algún tipo de tratamiento mediante un demonio, deje un delay de al menos 5 minutos entre cada petición que envíe.

5.2.9.2 Entrada

GetBookingRequest. Puede encontrar la documentación detallada en: http://javadocs.desarrollo.grupoiris.net/air/current/com/gi/ws/air/model/GetBookingRequest.html

Se debe debe especificar es el bookingId que se le ha devuelto en la operación de Book, o bien el localizador de la reserva que se quiere recuperar. En caso de que se especifiquen ambos valores, solo tendrá en cuenta el bookingId.

5.2.9.3 Salida

GetBookingReply. Puede encontrar la documentación detallada en: http://javadocs.desarrollo.grupoiris.net/air/current/com/gi/ws/air/model/GetBookingReply.html

El objeto contiene el mismo BookingInfo que en el proceso de reserva.

El estado de la reserva para saber si la reserva ya se ha completado en la compañía low cost, está accesible en `BookingInfo.lowcostStatus`. El valor `LowcostStatus.UNCONFIRMED` indica que la reserva está pendiente de confirmar por la compañía aérea.

5.2.9.4 Posibles errores

`InputErrorException` - Si hay algún error en la request enviada
`OperationErrorException` - Si se produce algún error en la cotización de la propuesta

5.2.10 cancel

5.2.10.1 Descripción

Permite cancelar una reserva.

Sólo se pueden cancelar reservas de vuelos regulares, es decir, aquellas que no son low cost.

5.2.10.2 Entrada

CancelRequest. La documentación ampliada está disponible en:
http://javadocs.desarrollo.grupoiris.net/air/current/com/gi/ws/air/model/CancelRequest.html

El único valor que debe especificar es el bookingId que se le ha devuelto en la operación de Book.

5.2.10.3 Salida

CancelReply. La documentación en línea se encuentra disponible en :
http://javadocs.desarrollo.grupoiris.net/air/current/com/gi/ws/air/model/CancelReply.html

Contiene un objeto BookingInfo con los datos de la reserva que se ha cancelado.

5.2.10.4 Posibles errores

- InputErrorException - Si hay algún error en la request enviada
- OperationErrorException - Si se produce algún error en la ejecución de la cancelación

5.2.11 getPNR

5.2.11.1 Descripción

Obtiene el PNR[ii] de una reserva tal y como está actualmente en el GDS[i].

Las reservas pueden ser modificadas entre el momento de reserva y el momento de emisión, por lo que esta operación nos devuelve el estado actual.

Las reservas de LowCost no tienen un proceso de emisión posterior a la reserva, por lo que esta operación sólo devuelve información de las reservas con vuelos regulares no LowCost.

5.2.11.2 Entrada

GetPNRRequest. Contiene un único campo "bookingId" en el que debe especificar el identificador devuelto en el proceso de reserva. Puede ver la documentación en línea en
http://javadocs.desarrollo.grupoiris.net/air/current/com/gi/ws/air/model/GetPNRRequest.html

5.2.11.3 Salida

GetPNRReply. La documentación en línea está disponible en
http://javadocs.desarrollo.grupoiris.net/air/current/com/gi/ws/air/model/GetPNRReply.html

El objeto devuelto contiene un PNRInfo con los datos guardados en el GDS.

El PNRInfo contiene una lista de PNRBookingInfo. Si la reserva es una combinación de dos reservas de sólo ida, la lista tendrá dos elementos. Si por el contrario, la reserva es normal, sólo contiene uno.

Cada uno de los objetos PNRBookingInfo contiene la siguiente información:

- bookingPrice: el precio total de todos los pasajeros y que será cargado en la forma de pago indicada en el proceso de emisión.
- Locator: El identificador de la reserva en el GDS[i].
- PlattingCarrier: La compañía con la que se realiza la emisión.
- Status: estado de la reserva. Normalmente el status de todos los elementos es el mismo. Sólo en el caso de que el agente de viajes emita parte de una reserva, puede ser distinto entre dos pasajeros.
- PaxInfo: Lista de PNRPaxInfo, un elemento por pasajero.

Los agentes de viajes pueden modificar las reservas por terminal, realizando modificaciones de todo tipo, pudiendo hacer, por ejemplo, que un pasajero tenga unos segmentos y otro otros distintos. Debido a esto, el modelo utilizado para devolver los datos, es diferente al utilizado en

el proceso de reserva. Mientras que en el proceso de reserva todos los pasajeros tienen el mismo trayecto, aquí la información va pasajero a pasajero, indicando por cada uno de ellos, todos los precios y los segmentos asociados. Esta información está contenida en el objeto PNRPaxInfo:

- endorsement: Texto de endosos que aparece en el billete. En principio este texto sólo se devuelve una vez que el billete esté emitido.
- Pax: de tipo PaxInfo ya descrito en la respuesta de la reserva. La información completa sobre este objeto está disponible en http://javadocs.desarrollo.grupoiris.net/air/current/com/gi/ws/air/model/PaxInfo.html
- segment: Lista de PNRSegment. Contiene la misma información que Segment (usado durante el proceso de reserva), a la que se le han añadido los siguientes datos:
  - baggageInfo: Información sobre el equipaje permitido tal y como lo devuelve el GDS.
  - FareBasis: Código de la tarifa aplicada para el segmento.
  - NotValidBefore/notValidAfter: dependiendo de la tarifa, puede ser posible realizar cambios en los segmentos (por ejemplo, para adelantarlo o atrasarlo). Estas fechas indican la fecha mínima y máxima que es posible modificar la fecha del segmento. NOTA: los cambios suelen tener coste para el cliente.
- TotalPrice: El precio total con todos sus datos de este pasajero.
- TicketNumber: Aquí se indica el número (o números) de billete una vez que se ha realizado el proceso de emisión.

5.2.11.4 Posibles errores

- `InputErrorException` - Si hay algún error en la request enviada
- `OperationErrorException` - Si se produce algún error en la obtención del PNR de la reserva

5.2.12 ticket

5.2.12.1 Descripción

Realiza la emisión electrónica de la reserva que se ha consultado previamente, en la misma sesión con getPNR. Sólo se emiten las reservas de vuelos regulares de GDS, no se pueden emitir las reservas LowCost.

Muy importante: cualquier billete emitido y no cancelado en el día de emisión, no podrá ser cancelado a posteriori, siendo necesario que la agencia de viajes realice un reembolso que suele tener penalizaciones.

Por favor, revise cualquier error que pueda ocurrir en el desarrollo de esta funcionalidad y asegúrese de que ha cancelado todos los billetes que haga de prueba.

Puede acceder al interfaz normal del sistema con el mismo usuario y clave que tiene para acceder al WebService y consultar desde este interfaz el estado de las reservas. Si aún así tiene dudas, póngase en contacto con la agencia de viajes para que ellos puedan revisar la reserva y cancelar los billetes antes de que sea demasiado tarde.

5.2.12.2 Entrada

TicketRequest. La documentación completa está disponible en http://javadocs.desarrollo.grupoiris.net/air/current/com/gi/ws/air/model/TicketRequest.html

Datos:

- paymentMode. Indica la forma de pago con la que la agencia realiza el pago al GDS, con los siguientes datos:
  - type: forma de pago. Puede ser CASH (la agencia de viajes paga a la compañía aérea en metálico) o CREDIT_CARD (la agencia de viajes paga a la compañía aérea con la tarjeta especificada en "creditCard".
  - CreditCard: Datos de la tarjeta de crédito que se utilizará para que la agencia de viajes pague el billete a la compañía aérea. Sólo en caso de que type tenga el valor "CREDIT_CARD". Los datos a enviar son: http://javadocs.desarrollo.grupoiris.net/air/current/com/gi/ws/air/model/CreditCard.html

NOTA SOBRE LA FORMA DE PAGO:

La forma de pago indicada aquí, no tiene porqué ser la forma de pago con la que el cliente final paga la reserva. Aquí se envía cómo paga la agencia que tiene el acceso a Amadeus, la reserva a la compañía aérea.

Por ejemplo, si un cliente final puede pagar la reserva con su tarjeta de crédito a través de un TPV, la forma de pago que se envía aquí no puede ser la misma tarjeta ya que en caso de ser así, estaríamos cobrando dos veces al cliente.

5.2.12.3 Salida

TicketReply. La documentación completa está disponible en http://javadocs.desarrollo.grupoiris.net/air/current/com/gi/ws/air/model/TicketReply.html

La respuesta contiene un único objeto PNRInfo idéntico al descrito en 5.2.11.3 Salida.

5.2.12.4 Posibles errores

`InputErrorException` - Si hay algún error en la request enviada
`OperationErrorException` - Si la operación de emisión falla

5.2.13 ticketCancel

5.2.13.1 Descripción

Cancela todos los billetes de una reserva.

Los GDS[i] sólo permiten la cancelación de billetes emitidos en el día. Cualquier intento de cancelar los billetes de una reserva emitida en un día diferente al actual devuelve un error de GDS.

Muy importante: cualquier billete emitido y no cancelado en el día de emisión, no podrá ser cancelado a posteriori, siendo necesario que la agencia de viajes realice un reembolso que suele tener penalizaciones.

Por favor, revise cualquier error que pueda ocurrir en el desarrollo de esta funcionalidad y asegúrese de que ha cancelado todos los billetes que haga de prueba.

Puede acceder al interfaz normal del sistema con el mismo usuario y clave que tiene para acceder al WebService y consultar desde este interfaz el estado de las reservas. Si aún así tiene dudas, póngase en contacto con la agencia de viajes para que ellos puedan revisar la reserva y cancelar los billetes antes de que sea demasiado tarde.

5.2.13.2 Entrada

TicketCancelRequest. La información completa está disponible en

http://javadocs.desarrollo.grupoiris.net/air/current/com/gi/ws/air/model/TicketCancelRequest.html

Sólo lleva un parámetro, el campo bookingId donde se indica el identificador de la reserva a la que se quiere cancelar los billetes.

5.2.13.3 Posibles errores

`InputErrorException` - Si hay algún error en la request enviada
`OperationErrorException` - Si la cancelación de billetes falla.

5.2.14 saveExternalPNR

5.2.14.1 Descripción

Guarda el PNR de una reserva externa en el sistema. Posteriormente, la reserva puede ser recuperada mediante la llamada a getPNR.

Esta operación solo es válida sobre reservas de Amadeus.

5.2.14.2 Entrada

SaveExternalPNRRequest. La información completa está disponible en http://javadocs.desarrollo.grupoiris.net/air/current/com/gi/ws/air/model/SaveExternalPNRRequest.html

5.2.14.3 Salida

SaveExternalPNRReply. La documentación en línea está disponible en http://javadocs.desarrollo.grupoiris.net/air/current/com/gi/ws/air/model/SaveExternalPNRReply.html

El objeto devuelto contiene un "bookingId" con el identificador devuelto de la reserva guardada. Este "bookingId" se usa de entrada en la llamada a getPNR para obtener el PNR de la reserva.

5.2.14.4 Posibles errores

`InputErrorException` - Si hay algún error en la request enviada
`OperationErrorException` - Si la operación falla

6 Proceso de reserva

6.1 Introducción

La mayor prioridad en la creación de estos WebServices ha sido el simplificar al máximo la implementación de un sistema estándar de reservas.

Para que sea lo más simple posible, están desarrollados utilizando sesión, es decir, durante el proceso se tiene en cuenta las operaciones lanzadas previamente, reduciendo así el número de datos que tienen que ser enviados para realizar cualquier operación.

6.2 Login

Antes de poder realizar cualquier operación en el sistema es necesario realizar el proceso de login. Este proceso nos devuelve el "endpoint" que nos da acceso al servicio de reserva.

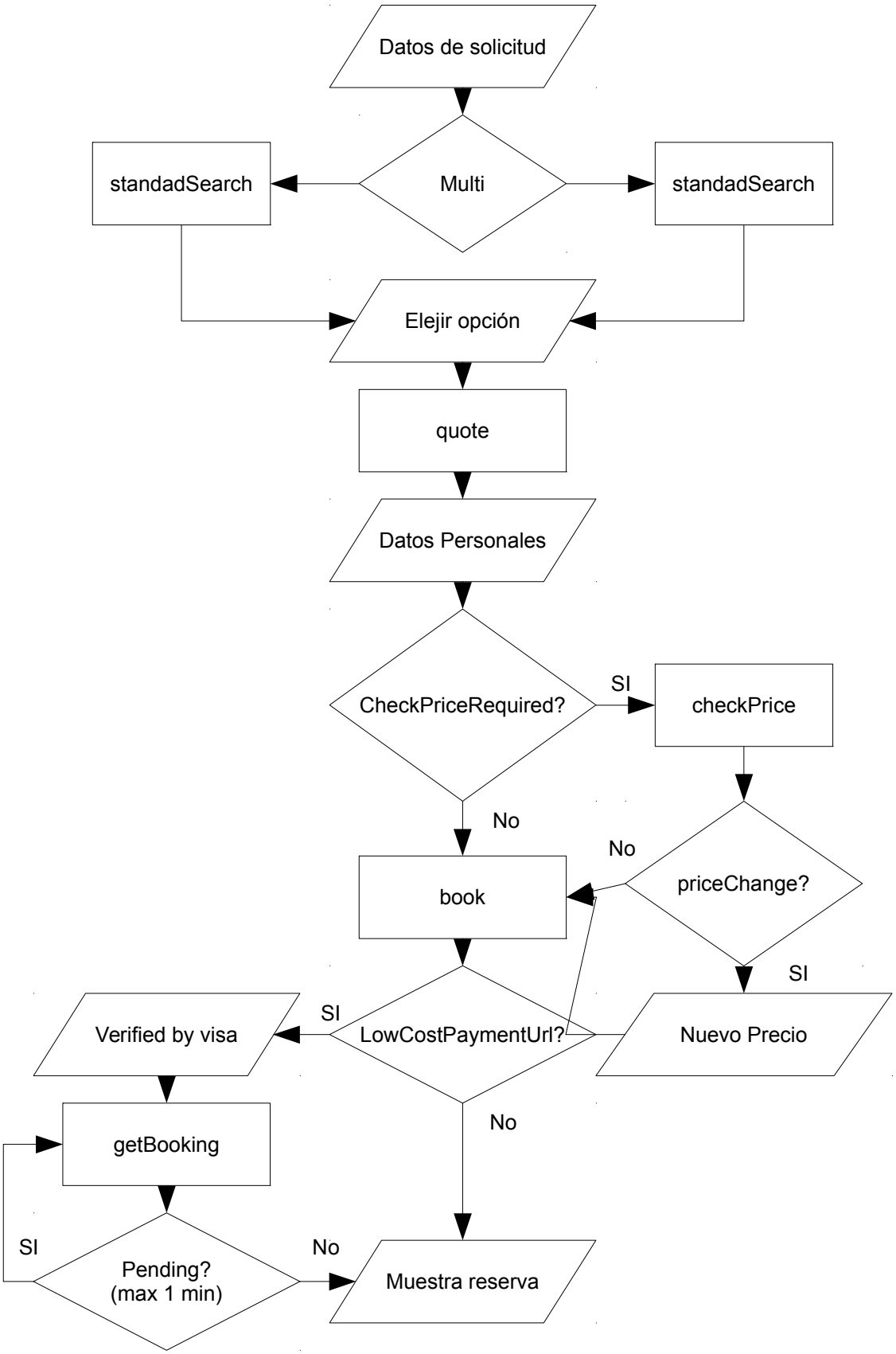
6.3 Flujo del proceso de reserva

Una vez realizado login, el flujo del proceso de reserva viene indicado por el diagrama que viene a continuación, donde:


Representa una interacción con el usuario


Representa una llamada al WS


Representa una decisión.


Datos de solicitud
Multi
standadSearch
standadSearch
Elejir opción
quote
Datos Personales
CheckPriceRequired?
SI
checkPrice
No
priceChange?
No
book
SI
LowCostPaymentUrl?
SI
Verified by visa
Nuevo Precio
No
getBooking
SI
Pending?
(max 1 min)
No
Muestra reserva

No es posible saltar ningún paso pero sí podemos volver hacia atrás siempre que queramos, es decir, no es posible llamar al método que ofrece el servicio de cotización sin antes haber realizado una petición de disponibilidad, pero sí que podemos reiniciar el proceso desde cualquier punto.

Siguiendo este flujo, el sistema estará preparado para reservas tanto de Amadeus, como de low cost, como de combinaciones de dos tarifas.

Si introduce una TPV, deberá introducirla justo antes de realizar la llamada a la operación de book.

7 Proceso de emisión

7.1 Introducción

En el aéreo, la reserva sólo garantiza la plaza en la clase elegida hasta que llegue la fecha límite de emisión. Para que el viajero pueda viajar, es necesario emitir los billetes.

Algunas agencias de viajes prefieren controlar ellas el proceso de emisión, realizándose siempre por terminal, si es así, no tienen que realizar ningún desarrollo de esta parte.

El proceso de emisión no funciona en el sistema de test de Amadeus, por lo que no es posible desarrollar sobre el test1, siendo necesario realizar este desarrollo utilizando un acceso REAL de Amadeus, en el test2.

Los billetes emitidos pueden cancelarse durante el día de emisión, pero NO es posible cancelarlos al día siguiente. Si un billete no se cancela en el día ya no se podrá cancelar, siendo necesario solicitar un reembolso. Dependiendo de la tarifa, el reembolso puede ser total (habitualmente sólo con tarifas muy caras), tener un coste o incluso el coste del reembolso ser el 100% del precio del billete. ES MUY IMPORTANTE que ante cualquier duda sobre si una reserva ha sido emitida o correctamente cancelado el billete, se ponga en contacto con su agente de viajes para que la revise y pueda actuar sobre ella para evitar gastos posteriores.

También puede entrar en el interfaz de venta estándar del sistema contra el que esté desarrollando y ver las reservas que ha realizado y el estado que tienen con su usuario y clave. La dirección de acceso del test2 es: http://test2.aereo.grupoiris.net/

Grupo Iris no se hará cargo de billetes que hayan quedado emitidos en el desarrollo del sistema.

7.2 Cambios en la reserva

Desde que se realiza la reserva hasta que se va a emitir, es posible que la reserva sea modificada externamente. Estos cambios pueden ser debidos a:

- Modificaciones realizadas por los agentes de viajes. Pueden ser por varios motivos tales como que el viajero quiera modificar la reserva o que el agente detecte alguna mejora en la reserva.
- Modificaciones realizadas por la compañía aérea.
- Modificaciones en la tarifa. Una vez reservada una determinada tarifa, las compañías mantienen el precio de la tarifa durante un periodo de tiempo fijo que depende del proveedor. Una vez transcurrido este periodo, es necesario eliminar la tarifa almacenada en la reserva y aplicar una nueva para poder emitir. En este proceso puede haber variaciones en el precio.

7.3 Fecha límite de emisión

El campo "lastTicketingDate" que se devuelve a la hora de realizar la reserva contiene la fecha límite de emisión que nos devuelve Amadeus en ese momento. Este fecha indica que si la reserva no se emite, una vez terminado el día especificado, la reserva es cancelada automáticamente por la compañía aérea.

Las compañías pueden modificar esta fecha límite de emisión a posteriori. Cuando esto sucede, la agencia de viajes recibe un mensaje en las colas de su terminal. RECUERDE QUE NO PUEDE IMPLEMENTAR NINGÚN TIPO DE DEMONIO NI PROCESO BATCH, por lo que debe ser el agente de viajes el que puede:

a) Emitir la reserva directamente en el terminal de Amadeus.

b) Recuperar la reserva en el interfaz estándar de la aplicación de Grupo Iris y emitirla

desde este interfaz.

c) Si así lo acuerda con el cliente y quiere que la emisión se realice desde su interfaz, tiene que darle un punto de entrada para que pueda iniciar el proceso de emisión.

7.4 Flujo del proceso de emisión

Antes de realizar cualquier llamada al WS para iniciar el proceso de emisión, debe realizar un login en el sistema, tal y como se indica en 5.1.1 login.

Debido a que las reservas pueden sufrir variaciones entre el momento en que se crea la reserva y el momento en el que se emite, para emitir una reserva es necesario realizar dos transacciones contra el sistema:

- Una para consultar el PNR[ii].
- Otra para solicitar la emisión.

Sólo es posible emitir reservas de compañías regulares, las reservas de compañías LowCost no se pueden emitir ya que no tienen un proceso de emisión diferenciado del de la reserva, por lo tanto, sólo se pueden emitir las reservas que tengan algún bookingInfo/providerBookingReference/status con valor "BOOKED".

8 Casos especiales

8.1 Residentes

Los ciudadanos residentes en Canarias y Baleares, tienen derecho a un descuento del 50% sobre el importe de la tarifa y de las tasas siempre y cuando estén realizando un viaje de ida o ida y vuelta entre la península y su Isla de residencia.

En http://www.caib.es/sacmicrofront/contenido.do?mkey=M169&lang=ES&cont=3903 vienen los documentos con las normas que rigen este descuento.

Las compañías low cost no ofrecen este descuento, por lo que el sistema no busca en low cost cuando se introduce algún residente.

No es posible solicitar residentes en una petición de múltiples trayectos.

En la respuesta de cotización, cuando el viajero es residente, los campos residentOrLargeFamilyCredentialRequired y residentTownRequired dentro de QuoteReply / RequiredData / paxRequiredData son ambos true.

La única modificación sobre el proceso de reserva normal es que si el tipo de pasajero tiene los valores anteriores a true, tiene que solicitarle además:

- El tipo de identificación que aporta (uno de los indicados en http://javadocs.desarrollo.grupoiris.net/air/current/com/gi/ws/v010/air/model/ResidentOrLargeFamilyCredentialType.html).
- El residentTownId (tiene todos los valores disponibles en la QuoteReply).
- La identificación con DNI para DN.
- La identificación del pasajero con NIE para TR.
- La edad del pasajero para MR.

Es importante señalar que, debido a cambios recientes en la reglamentación, todos los pasajeros residentes deberán presentar un certificado de residencia en el embarque, sea cual sea su tipo de identificación.

Con esta cambio, el número de certificado de residencia (*certificate*) es siempre opcional en la reserva, ya que le será exigido al pasajero en el embarque.

8.2 Familia numerosa

El descuento de familia numerosa es aplicable a cualquier ruta doméstica en territorio español. Este descuento aplica en todos los conceptos del billete excepto sobre las tasas de aeropuerto. El descuento de familia numerosa es compatible con el descuento de residente

Para solicitar la aplicación del descuento de familia numerosa es necesario indicar en la búsqueda de vuelos el tipo de descuento de familia numerosa que se quiere aplicar (http://javadocs.desarrollo.grupoiris.net/air/current/com/gi/ws/v010/air/model/BaseSearchRequest.html#largeFamilyType)

En la respuesta de cotización, cuando se ha solicitado el descuento de familia numerosa en la búsqueda, el campo QuoteReply / largeFamilyCredentialRequired indicando que debe proporcionarse la comunidad autónoma y el número de documento correspondiente. También el campo residentOrLargeFamilyCredentialRequired dentro de QuoteReply / RequiredData / paxRequiredData es true, por lo que debe solicitarse a cada viajero el tipo de acreditación que aporta (DN/TR/MR).

Hay dos modificaciones sobre el proceso de reserva normal:

- A nivel del BookRequest hay que aportar los datos de:
  - Comunidad autónoma que expide el certificado (uno de los códigos devueltos en el QuoteReply)
  - El número de documento de familia numerosa
- A nivel de pasajero se requieren los siguientes datos:
  - El tipo de identificación que aporta (uno de los indicados en http://javadocs.desarrollo.grupoiris.net/air/current/com/gi/ws/v010/air/model/ResidentOrLargeFamilyCredentialType.html).
  - La identificación con DNI para DN.
  - La identificación del pasajero con NIE para TR.
  - La edad del pasajero para MR.

8.3 Compañías low cost

Siguiendo la lógica indicada en el Proceso de reserva, su aplicación estará preparada para que no tenga que hacer nada especial cuando llegue una propuesta de low cost.

Aún así, tenga en cuenta los siguientes detalles:

- Las reservas de low cost no se pueden cancelar.
- Las compañías cobran al cliente en su tarjeta de crédito, el campo noChargedPrice dentro del QuoteInfo, le indica cuanto le falta por cobrar.
- Las reservas pueden tardar horas en confirmarse. Si una reserva no se confirma en los primeros momentos, si lo desea puede implementar un demonio para consultarlas cada unos 10 minutos. Nuestro sistema lo hace y si ha introducido un email de contacto (para B2C debería obligarle a introducirlo), nuestro sistema le avisa de que ha cambiado su estado.

8.4 Securyty Flight Passenger Data

Desde Agosto de 2009 y por iniciativa del TSA (Transportation Security Administration) de USA, para viajar a determinados destinos (por ejemplo USA, Reino Unido o Cuba) es necesario aportar varios datos a fin de verificar la identificación de todos pasajeros de la reservas aéreas. Los datos requeridos para cada pasajero por el SFPD son los siguientes:

- Nombre y apellidos del pasajeros
- Tipo de documento identificativo: DNI, NIE, Pasaporte
- Número de documento
- País de emisión del documento
- Fecha de caducidad del documento
- Nacionalidad.
- Fecha de nacimiento

Cuando un país de destino exige los datos del SFPD, la etiqueta securityFlightPassengerDataRequired de la respuesta de la cotización toma el valor true. Cuando esto sucede, se deben enviar los datos requeridos bien el operación de reserva

cumplimentando los datos de los pasajeros o bien en la emisión mediante la etiqueta securityFlightPassengerData.

Si el GDS requiere los datos del SFPD y estos no son aportados ni en la reserva ni en la emisión, la operación de emisión lanzará un error de tipo SFPD_REQUIRED.

i Global Distribution System. Consulte http://en.wikipedia.org/wiki/Global_Distribution_System para obtener más información.

ii Passenger Name Record. Consulte http://en.wikipedia.org/wiki/Passenger_Name_Record para obtener más información.