PROYECTO TI
TRAZABILIDAD CONTENEDORES
-HERMES-

Manual De Integración De Almacenistas

Junio - 2026

CONTROL DEL DOCUMENTO

Información del Documento	
Identificación Documento	Manual De Integración De Almacenistas
Fecha	30.04.2026
Última Fecha Actualizado	25.06.2026

Historia del Documento

Versión	Fecha	Cambio	Autor
1.0	30.04.2026	Creación del documento.	SDI
1.1	08.05.2026	Modificación del documento.	SDI
1.2	17.06.2026	Se agregan IPs públicas de Aduana	SDI
1.3	25.06.2026	Se actualiza flujo de validación de suscripción	SDI

Aprobación del Documento

Rol	Nombre	Unidad	Fecha
Líder de Negocio			

Contenido

Introducción

Objetivo del Documento

Este manual tiene como objetivo proporcionar a los almacenistas las instrucciones técnicas necesarias para integrarse al sistema HERMES del Servicio Nacional de Aduanas, específicamente al servicio de notificaciones de bloqueo de marcas de manifiestos. A través de este documento, el almacenista podrá comprender y ejecutar el proceso completo de suscripción a la mensajería, así como preparar su infraestructura para la recepción de notificaciones de bloqueo en tiempo real.

Reseña

HERMES es el sistema de trazabilidad de contenedores del Servicio Nacional de Aduanas. Como parte de sus capacidades operativas, HERMES notifica a los almacenistas cuando un manifiesto marítimo y su BL asociado o un manifiesto terrestre presentan marcas de selección aduanera (como por ejemplo aforo, examen físico o revisión documental entre otras), permitiendo que los recintos de almacenamiento actúen oportunamente ante estas situaciones.

La integración se realiza mediante una API REST expuesta por HERMES, combinada con un mecanismo de publicación/suscripción basado en Amazon SNS. Para que un almacenista pueda recibir estas notificaciones, debe registrar y validar un endpoint HTTP(S) propio ante HERMES, proceso que se describe en detalle en las secciones siguientes.

Este documento está dirigido a los equipos técnicos de los almacenistas y asume conocimientos básicos en desarrollo de APIs REST y consumo de servicios HTTP.

Flujos de Integración

Flujo de Validación de Suscripción

Antes de que un almacenista pueda recibir notificaciones de bloqueo, HERMES requiere verificar que el almacenista tiene control efectivo sobre el endpoint que registra. Este proceso se denomina validación de suscripción y se ejecuta una única vez al momento del registro.

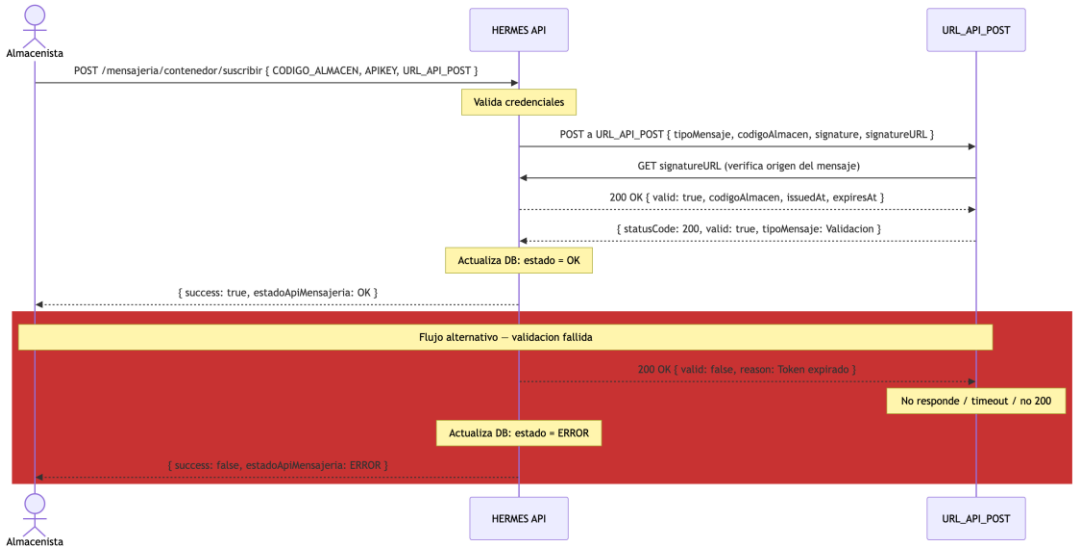
El flujo consta de los siguientes pasos:

1. Registro previo en HERMES: El almacenista debe contar con un registro activo en el sistema, el cual incluye un código de almacén (CODIGO_ALMACEN) y una API Key asignada. Sin este registro previo, no es posible iniciar la suscripción.
2. Solicitud de suscripción: El almacenista invoca el endpoint POST *https://api-hermes.aduana.cl/mensajeria/contenedor/suscribir* enviando su código de almacén, su API Key y la URL de su endpoint receptor (URL_API_POST). HERMES verifica las credenciales y rechaza la solicitud si no coinciden o el almacenista se encuentra inactivo.
3. Envío del mensaje de validación: Si las credenciales son válidas, HERMES realiza un POST a la URL_API_POST informada, incluyendo un token firmado (signature) y una URL de verificación (signatureURL).
4. Verificación del origen por parte del almacenista: El endpoint del almacenista debe realizar un GET al signatureURL para confirmar que el mensaje proviene efectivamente de HERMES. Este paso es obligatorio y protege al almacenista de suplantaciones.
5. Confirmación de validación: Si el GET retorna "valid": true, el almacenista responde al POST original con una confirmación exitosa. HERMES registra la URL como validada y actualiza el estado del almacenista a OK. El json que espera HERMES como respuesta es:

```
{
  "status": 200,
  "data": {
    "codigoAlmacen": "A00",  <aquí debe ir el codigo del Almacenista>
    "tipoMensaje": "Validacion",
    "valid": true
  },
  "message": "Validacion realizada"
}
```

6. Resultado del proceso: HERMES responde a la solicitud inicial confirmando que la suscripción fue registrada y validada correctamente. A partir de este momento, el endpoint quedará habilitado para recibir notificaciones de bloqueo.

Si en cualquier paso la validación falla (timeout, respuesta no 200, token inválido o expirado), el estado del almacenista se actualiza a ERROR y la URL no queda registrada. El almacenista deberá reintentar el proceso de suscripción desde el punto 2 en adelante.



Flujo de Bloqueo

Una vez que el almacenista tiene su suscripción validada y activa, HERMES notificará automáticamente cuando un bloqueo de marca asociado a un manifiesto reciba marcas de selección aduanera.

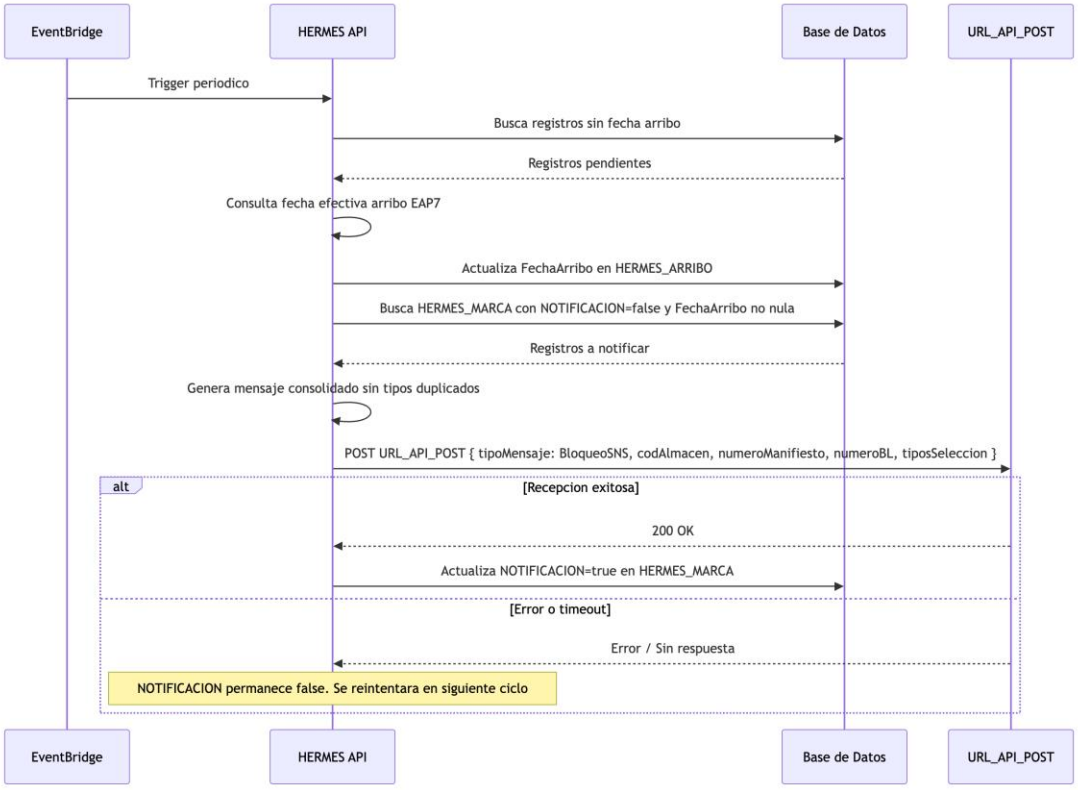
El proceso ocurre de forma transparente para el almacenista: HERMES monitorea periódicamente los registros pendientes de notificación. Cuando detecta una marca activa y fecha de arribo confirmada, genera el mensaje de bloqueo y lo envía al endpoint registrado por el almacenista durante la suscripción.

El mensaje de bloqueo llegará al URL_API_POST registrado con la siguiente estructura:

```json
{
  "tipoMensaje": "BloqueoSNS",
  "codAlmacen": "A00",
  "numeroManifiesto": "MFTO-2026-001",
  "numeroBL": "PHXA08500",
  "tiposSeleccion": ["AFORO", "EXAMEN FISICO", "REVISION DOCUMENTAL"]
}
```

El almacenista debe procesar este mensaje y actualizar su sistema interno con las marcas de selección informadas. Es importante considerar que tiposSeleccion nunca contendrá tipos duplicados: si un manifiesto tiene múltiples marcas del mismo tipo, HERMES las consolida enviando cada tipo una sola vez.

Una vez que HERMES recibe una recepción exitosa por parte del almacenista, marca internamente la notificación como enviada (NOTIFICACION = true), dejando trazabilidad del bloqueo para efectos de auditoría.



Implementación del lado del Almacenista de ejemplo

La implementación de referencia fue desarrollada con las siguientes tecnologías:

Capa	Tecnología	Versión
Runtime	Node.js	24.x
Framework	AdonisJS	7.0

Lenguaje	TypeScript	5.9.3
Validación	VineJS	4.3.0

Estructura del Endpoint Receptor

Se expone un único endpoint que recibe ambos tipos de mensaje (Validacion y BloqueoSNS):

```
POST /api/v1/ejemplos/almacenista/recepcion-bloqueo

Content-Type: application/json
```

El enrutado se define en start/routes.ts:

```
router
 .group(() => {
 router.post('/ejemplos/almacenista/recepcion-bloqueo',
 [RecepcionBloqueoController, 'store'])
 })
 .prefix('/api/v1')
```

El prefijo /api/v1 agrupa todos los endpoints bajo un versionado común. El endpoint no requiere autenticación dado que los mensajes son enviados por HERMES y se autentican mediante el mecanismo de firma en el flujo de validación.

Tipos e Interfaces Todas las estructuras de datos están tipadas en app/interfaces/recepcion_bloqueo.ts:

```
export type TipoMensajeRecepcion = 'Validacion' | 'BloqueoSNS'

// Payload para mensajes de validación de suscripción
export interface ValidacionPayload {
 tipoMensaje: 'Validacion'
 codigoAlmacen?: string
 signature?: string
 signatureURL?: string
}

// Payload para mensajes de bloqueo de BL
export interface BloqueoSnsPayload {
 tipoMensaje: 'BloqueoSNS'
 codAlmacen?: string
 numeroManifiesto?: string
 numeroBL?: string
 tiposSeleccion?: string[]
}
```

```
// Respuesta de HERMES al verificar la firma
export interface VerificacionAduanaResponse {
 valid: boolean
 reason?: string
 codigoAlmacen?: string
}
```

TypeScript permite definir contratos explícitos entre los componentes, eliminando errores de campos mal nombrados o tipos incorrectos en tiempo de compilación.

Validación de Entrada

El validador (app/validators/recepcion_bloqueo.ts) utiliza VineJS para validar la estructura del cuerpo de la petición antes de que llegue al controlador:

```
import vine from '@vinejs/vine'

export const storeRecepcionBloqueo = vine.create(
 vine.object({
  tipoMensaje: vine.string().trim(), // Requerido: discrimina el flujo
  codigoAlmacen: vine.string().trim().optional(), // Para Validacion
  signature: vine.string().trim().optional(), // Para Validacion
  signatureURL: vine.string().trim().url().optional(), // Para Validacion — validado como URL
  codAlmacen: vine.string().trim().optional(), // Para BloqueoSNS
  numeroManifiesto: vine.string().trim().optional(), // Para BloqueoSNS
  numeroBL: vine.string().trim().optional(), // Para BloqueoSNS
  tiposSeleccion: vine.array(vine.string().trim()).optional(), // Para  BloqueoSNS
 })
)
```

El campo tipoMensaje es el único requerido en esta capa. El resto de las validaciones de negocio (campos obligatorios según tipo) se realizan en la capa de servicio. Si signatureURL está presente, VineJS valida que tenga formato de URL válida antes de que el sistema intente hacer una petición a esa dirección.

Controlador

El controlador (app/controllers/recepcion_bloqueo_controller.ts) es el punto de entrada HTTP. Su responsabilidad es validar la entrada, delegar al servicio y formatear la respuesta:

```
export default class RecepcionBloqueoController {
  async store(ctx: HttpContext) {
    try {
      // 1. Valida el cuerpo de la petición con el esquema VineJS
      const payload = await ctx.request.validateUsing(storeRecepcionBloqueo)

      // 2. Delega la lógica al servicio
      const resultado = await RecepcionBloqueoService.procesarMensaje(payload)

      // 3. Registra el resultado según severidad
      if (resultado.statusCode >= 500) {
        console.error('❌ Error de procesamiento.', {
          statusCode: resultado.statusCode
        })
      } else if (resultado.statusCode >= 400) {
        console.warn('⚠️ Mensaje rechazado.', {
          statusCode: resultado.statusCode
        })
      } else {
        console.log('✅ Mensaje procesado.', {
          statusCode: resultado.statusCode
        })
      }

// 4. Responde con el status code apropiado y estructura estándar
      return ctx.response
        .status(resultado.statusCode)
        .send(crearRespuesta(
          resultado.statusCode,
          resultado.data,
          resultado.message)
        )
    } catch (error) {
      console.error('❌ Error inesperado:', error)
      throw error
    }
  }
}
```

El controlador no contiene lógica de negocio. Si la validación VineJS falla (por ejemplo, signatureURL no es una URL válida), AdonisJS retorna automáticamente un 422 Unprocessable Entity antes de que el controlador sea invocado.

Servicio — Lógica de Procesamiento

El servicio (app/services/recepcion_bloqueo_service.ts) contiene toda la lógica de negocio. El método procesarMensaje actúa como dispatcher según tipoMensaje:

```
static async procesarMensaje(payload: RecepcionBloqueoRequest) {
  if (payload.tipoMensaje === 'Validacion') {
    return this.procesarValidacion({ ...payload, tipoMensaje: 'Validacion' })
  }
  if (payload.tipoMensaje === 'BloqueoSNS') {
    return this.procesarBloqueo({ ...payload, tipoMensaje: 'BloqueoSNS' })
  }
  return { statusCode: 400, data: null, message: MENSAJES.TIPO_NO_RECONOCIDO }
}
```

Procesamiento de Validación

```
static async procesarValidacion(payload: ValidacionPayload) {
// Verifica campos requeridos
  if (!payload.signature) {
    return {
      statusCode: 400,
      data: null,
      message: MENSAJES.SIGNATURE_REQUERIDO
    }
  }

  if (!payload.signatureURL) {
    return {
      statusCode: 400,
      data: null,
      message: MENSAJES.SIGNATURE_URL_REQUERIDO
    }
  }

  try {
    // Consulta a HERMES para verificar que la firma es auténtica
    const verificacion = await this.verificarConAduana(payload.signatureURL)

    if (!verificacion.valid) {
      return {
        statusCode: 401,
        data: {
          codigoAlmacen: verificacion.codigoAlmacen,
          reason: verificacion.reason,
          valid: false
        },
        message: `${MENSAJES.FIRMA_INVALIDA}: ${verificacion.reason}`,
      }
```

```
  }

  return {
    statusCode: 200,
    data: {
      codigoAlmacen: verificacion.codigoAlmacen,
      tipoMensaje: 'Validacion',
      valid: true
    },
    message: MENSAJES.VALIDACION_REALIZADA,
  }
 } catch (error) {
  return {
    statusCode: 500,
    data: {
      detalle: error instanceof Error ? error.message : 'Error desconocido'
    },
    message: MENSAJES.ERROR_VERIFICACION_ADUANA,
 }
 }
}
```

Verificación de Firma con HERMES

Este método realiza el GET al signatureURL provisto por HERMES para confirmar la autenticidad del mensaje. El timeout es configurable mediante la variable de entorno ADUANA_SIGNATURE_TIMEOUT_MS (por defecto 10 segundos):

```
static async verificarConAduana(signatureURL: string): Promise<VerificacionAduanaResponse> {
 const timeout = env.get('ADUANA_SIGNATURE_TIMEOUT_MS') ?? 10000

// AbortSignal.timeout cancela automáticamente si HERMES no responde a tiempo
 const response = await fetch(signatureURL, {
 signal: AbortSignal.timeout(timeout),
})

 if (!response.ok) {
  throw new Error(
   `Servicio ADUANA respondio con estatus ${response.status}`
  )
 }

 const payload = await response.json() as Partial<VerificacionAduanaResponse>

 if (typeof payload.valid !== 'boolean') {
  throw new Error('Respuesta invalida del servicio de ADUANA')
 }
```

```
 return {
  valid: payload.valid,
  reason: payload.reason,
  codigoAlmacen: payload.codigoAlmacen
 }
}
```

Si la petición al signatureURL excede el timeout o retorna un status no exitoso, el método lanza una excepción que procesarValidacion captura y convierte en respuesta 500.

Procesamiento de BloqueoSNS

```
static async procesarBloqueo(payload: BloqueoSnsPayload) {
// Valida que todos los campos del bloqueo estén presentes
 if (
  !payload.codAlmacen ||
  !payload.numeroManifiesto
 {
   return {
    statusCode: 400,
    data: null,
    message: MENSAJES.PAYLOAD_BLOQUEO_INCOMPLETO }
 }
 if (
  !payload.tiposSeleccion ||
  payload.tiposSeleccion.length === 0
 ) {
  return {
   statusCode: 400,
   data: null,
   message: MENSAJES.TIPOS_SELECCION_REQUERIDO }
 }

// Acuse de recibo exitoso — aquí el almacenista debe agregar su lógica de negocio
 return {
  statusCode: 200,
  data: {
   codAlmacen: payload.codAlmacen,
   numeroManifiesto: payload.numeroManifiesto,
   numeroBL: payload.numeroBL,
   tiposSeleccion: payload.tiposSeleccion,
   timestamp: new Date().toISOString(),
  },
  message: MENSAJES.BLOQUEO_RECIBIDO,
 }
}
```

El método valida que el payload esté completo y retorna un acuse de recibo con timestamp. En una implementación productiva, entre la validación y el retorno de la respuesta el almacenista debe insertar la lógica que actualice su sistema interno (base de datos, cola de mensajes, etc.).

Tabla de Respuestas

Escenario	**HTTP Status**	**message**
Validación exitosa	200	Validacion realizada
BloqueoSNS recibido	200	Mensaje de bloqueo recibido
Falta signature o signatureURL	400	signature requerido / signatureURL requerido
Payload de bloqueo incompleto	400	Payload de BloqueoSNS incompleto
tiposSeleccion vacío	400	tiposSeleccion requerido
tipoMensaje no reconocido	400	Tipo de mensaje no reconocido
Firma inválida según HERMES	401	Firma invalida: <reason>
Error de red con HERMES	500	Error al verificar firma con ADUANA

Anexo – Tipos de Selección a informar

Se adjuntan los codigos de tipo selección que seran informados:

2	AFORO
3	EXAMEN FISICO
4	REVISION DOCUMENTAL
5	ESCANER
6	REVISION BULTOS
7	REVISION VEHICULO
8	REVISION DE PRECINTOS Y SELLOS
9	TOMA DE MUESTRAS
10	REVISION DE CABINA

Anexo – Habilitación y Pruebas

Procedimiento ambiente pruebas

Cada participante debe completar la siguiente información y enviarla por correo:

Asunto Correo: [Hermes] Integracion Almacenistas - Habilitación Ambiente Pruebas

Nombre empresa:
Rut Empresa:
Domicilio Empresa
Representante Legal:
(*) Tipo Participante:
Código participante:
Ambiente: Pruebas/Producción (En esta etapa sólo aplica Pruebas).

(*) Revisar en documento de especificación el tipo de almacenista que corresponda.

Una vez completada esta información, se deberá enviar al siguiente correo: proyectohermes@aduana.cl.

En cuanto se verifique la información completada, se compartirán y habilitaran las credenciales.

Procedimiento ambiente Producción

Una vez cumplido el hito de pruebas, el participante quedará autorizado y deberá enviar un correo con la siguiente información:

Asunto Correo: [Hermes] Integracion Almacenistas - Habilitación Ambiente Productivo

Nombre empresa:
Rut Empresa:
Domicilio Empresa
Representante Legal:
Tipo Participante:
Código participante:
Ambiente: Pruebas/Producción (En esta etapa sólo aplica Producción).

Una vez completada esta información, se deberá enviar al siguiente correo: proyectohermes@aduana.cl.

Anexo – IPs Publicas de Aduana

Se indican cuales son las IPs públicas de Aduana por si necesitan tenerlas para la administración de los firewalls de los Almacenistas.

Listado de IPs

164.77.213.91

190.208.62.234

44.222.37.126

44.217.130.152