Dossie de Integracao

Cameras Uniview - LiteAPI (IPC V5.04) - Integracao em Go

Objetivo

Consolidar os pontos essenciais para integrar uma camera Uniview via LiteAPI em um sistema escrito em Go: autenticacao (HTTP Digest), descoberta de capacidades/analiticos, criacao e manutencao de subscricao de eventos, recepcao/confirmacao (ACK) de notificacoes e um plano de testes para capturar e catalogar todos os analiticos suportados.

Escopo

- Comunicacao HTTP com a camera (LiteAPI) e RTSP apenas como contexto.
- Subscricao e recepcao de eventos/alarme (Notification).
- Como assinar todos os tipos de eventos de uma vez (Type bitmask) e como restringir por categoria.
- Como correlacionar eventos com analiticos (por URL e/ou AlarmType).
- Checklist e plano de testes para validar ponta-a-ponta.

Fonte

Baseado no documento "LiteAPI Document for IPC V5.04" (Zhejiang Uniview Technologies Co., Ltd, 2025-04-18). Secoes principais: 3.1 (HTTP/Digest), 5.12-5.15 (Smart/Face/Subscribe), 6.3 (Events), 6.9 (Smart/Capabilities) e 7.3 (Alarm Types).

1. Visao geral da arquitetura

A integracao se divide em dois papeis no seu sistema Go:

- Cliente HTTP (Go - outbound): chama a LiteAPI na camera (criar/renovar/excluir subscricao, consultar capacidades, configurar analiticos).
- Servidor HTTP (Go - inbound): recebe POSTs de notificacao (eventos/alarme) enviados pela camera e responde com um ACK (JSON) para confirmar o recebimento.

Fluxo resumido

Etapa	Direcao	Endpoints principais
(1) Descobrir capacidades	Go -> Camera	`GET /LAPI/V1.0/Channels/<ID>/Smart/Capabilities` `GET /LAPI/V1.0/Channels/<ID>/Smart/WorkingStatus`
(2) Criar subscricao	Go -> Camera	`POST /LAPI/V1.0/System/Event/Subscription` `(body com IP/porta do seu servidor)`
(3) Receber eventos	Camera -> Go	`POST /LAPI/V1.0/System/Event/Notification/...` `(ex.: /Alarm, /HatDetection, /PersonInfo, /Loitering)`
(4) Manter viva	Go -> Camera	`PUT /LAPI/V1.0/System/Event/Subscription/<ID>` `(keepalive)`
(5) Encerrar	Go -> Camera	`DELETE /LAPI/V1.0/System/Event/Subscription/<ID>`

Requisitos de rede

A camera precisa conseguir abrir conexao ate o IP/porta do seu servidor de eventos. Em geral, isso requer rotas/ACLs permitindo trafego de entrada para a porta do seu listener (ex.: 50235). Mantenha em mente NAT e firewalls quando o servidor nao estiver na mesma LAN.

2. HTTP e autenticacao (Digest)

A LiteAPI usa HTTP e trafega JSON. O padrao de URL segue o formato http(s)://<IP>:<Port>/<URL-da-interface>. A autenticacao e HTTP Digest (RFC2617).

Implementacao pratica em Go

- Use um cliente HTTP que suporte Digest: primeira chamada recebe 401 com WWW-Authenticate, depois reenvie com Authorization calculado (MD5).
- Defina timeouts (dial + response header) e retries com backoff para chamadas criticas (subscricao/keepalive).
- Nao embuta credenciais em codigo; use variaveis de ambiente/secret manager.

Headers recomendados

```
Content-Type: application/json
Host: <cameraIP>:<port>
Connection: close (opcional)
```

3. Subscricao de eventos (Event Subscription)

Criacao: POST /LAPI/V1.0/System/Event/Subscription. O corpo define para onde a camera deve enviar os eventos e por quanto tempo a subscricao e valida.

3.1 Criar subscricao (POST)

Campo	Descricao (resumo)
AddressType	Tipo de endereco. Em geral use 0 (IPv4).
IPAddress	IP do seu servidor receptor.
Port	Porta TCP do seu servidor receptor.
Duration	Validade (s). Intervalo tipico 30..3600.
Type (bitmask)	Categoria(s) de eventos para assinar (ver abaixo).
ImagePushMode	Como imagens sao enviadas: Base64/binario (0), URL local (1), URL cloud (2).

Exemplo de request (ajuste IP/porta)

```
{
  "AddressType": 0,
  "IPAddress": "<SEU_IP>",
  "Port": 50235,
  "Duration": 300,
  "Type": 8,
  "ImagePushMode": 0
}
```

3.2 Type (bitmask): assinar todos ou por categoria

O campo Type e um bitmask. Cada bit habilita uma categoria de subscricao. Exemplos comuns:

Categoria	Bit	Decimal
Device status type alarm	0	1
Monitoring service alarm	1	2
Pan-intelligent alarm	2	4
Smart alarm	3	8
Face recognition	4	16
Structured data	5	32
License plate recognition	6	64
ESP data	8	256
Person verification	10	1024
Vehicle traffic data	11	2048

Fire point detection alarms	12	4096
Alarm picture data	13	8192
People count	14	16384
Heat map	16	65536

Para assinar varias categorias, some os valores decimais (OR bit-a-bit). Para assinar as categorias acima (todas listadas), a soma e 97663.

3.3 Resposta da criacao

A resposta inclui um ID (usado no keepalive e no delete) e um campo Reference (com /Subscription/Subscribers/<ID>). Guarde ambos.

3.4 Keepalive (PUT)

Renove antes de expirar (ex.: a cada Duration/2). Interface: PUT /LAPI/V1.0/System/Event/Subscription/<ID>. Corpo: {"Duration": } (Reference pode ser opcional).

3.5 Excluir subscricao (DELETE)

Quando encerrar, delete: DELETE /LAPI/V1.0/System/Event/Subscription/<ID>.

4. Recepcao de eventos (Notification) e ACK

Quando um evento/alarme ocorre, a camera envia um POST para o seu servidor em um endpoint do tipo /LAPI/V1.0/System/Event/Notification/<Evento>. Exemplos: /Alarm, /HatDetection, /PersonInfo, /Loitering, etc.

4.1 Estrutura comum de notificacao

Quase todos os eventos incluem:

- Reference: URL de push (inclui o SubscribersID) usada para distinguir assinantes.
- AlarmType: string que identifica o tipo do alarme/analitico (lista completa em 7.3 Alarm Types).
- TimeStamp: UTC em segundos.
- Campos opcionais: Seq/AlarmSeq, SourceID/AlarmSrcID, RelatedID (ID global do evento na camera), etc.

Exemplo (Alarm Events)

```
{
  "Reference": "<ip>:<port>/Subscription/Subscribers/1",
  "AlarmInfo": {
    "AlarmType": "MotionAlarmOn",
    "AlarmLevel": 0,
    "TimeStamp": 1489040894,
    "AlarmSeq": 327,
    "AlarmSrcID": 0,
    "RelatedID": "5ED9FE4C0000001",
    "DeviceID": "..."
  }
}
```

4.2 ACK: como responder para confirmar recebimento

A camera espera um JSON de resposta (Response object) indicando sucesso. Use ResponseCode=0 e ResponseURL igual ao caminho recebido.

```
{
  "Response": {
    "ResponseURL": "/LAPI/V1.0/System/Event/Notification/Alarm",
    "ResponseCode": 0,
    "SubResponseCode": 0,
    "ResponseString": "Succeed",
    "StatusCode": 0,
    "Data": null
  }
}
```

4.3 Imagens nos eventos

Alguns eventos trazem imagens (AlarmPicture/ImageList). Dependendo do ImagePushMode, voce pode receber Base64/binario ou URLs (ex.: /LAPI/V1.0/System/Picture?Type=...&Index;=...&Size;=...). Ao receber URL, faca um GET autenticado (Digest) para baixar a imagem.

5. Descoberta e habilitacao de analiticos

Para saber o que a camera suporta (e entao testar/capturar), use a consulta de capacidades e, quando necessario, habilite o recurso e configure regras/areas/agenda.

5.1 Consultar capacidades por canal

```
GET /LAPI/V1.0/Channels/<ID>/Smart/Capabilities
```

Esse endpoint retorna um grande objeto com flags/limites de recursos (por exemplo: FaceDetection, IntrusionDetection, CrossLineDetection, AreasPeopleCounting, CrowdDensity, etc.).

5.2 Status global de smart function

Verifique/ajuste se funcoes smart estao habilitadas:

```
GET  /LAPI/V1.0/Channels/<ID>/Smart/WorkingStatus
PUT  /LAPI/V1.0/Channels/<ID>/Smart/WorkingStatus
```

5.3 Exemplos de configuracao (Smart)

Alguns analiticos requerem configurar regra, areas, plano semanal e acoes de linkage. Exemplos:

- Intrusion Detection: /Channels//Smart/IntrusionDetection/Rule, /Areas, /WeekPlan, /LinkageActions
- Cross Line Detection: /Channels//Smart/CrossLineDetection/Rule, /Areas, /WeekPlan, /LinkageActions
- Access/Leave Zone: /Channels//Smart/AccessZone/... e /Smart/LeaveZone/... (Rule/Areas/WeekPlan/LinkageActions)

Face recognition (alto nivel)

O documento descreve um processo tipico: habilitar face capture/compare, criar biblioteca de pessoas, cadastrar pessoas, habilitar monitoramento e entao buscar dados ou assinar eventos.

6. Plano de testes para capturar todos os analiticos

Objetivo: catalogar exatamente quais eventos a camera envia, com quais campos, e como eles se relacionam com os analiticos habilitados.

1. Preparar receiver: servidor HTTP que aceita qualquer rota em /LAPI/V1.0/System/Event/Notification/ e grava (path, headers, body) com timestamp.
2. Descobrir capacidades: GET /LAPI/V1.0/Channels/<ID>/Smart/Capabilities e salvar JSON (baseline).
3. Habilitar smart function: GET/PUT /LAPI/V1.0/Channels/<ID>/Smart/WorkingStatus conforme necessario.
4. Assinar por categoria: comece com Type=8 (Smart alarm), depois adicione Face (16), People count (16384), etc.
5. Gerar eventos: acione os analiticos (ex.: cruzar linha, entrar em area, loitering) e confirme que chegam no receiver.
6. Validar imagens: teste ImagePushMode=0 e ImagePushMode=1 (URL local) e baixe imagens via GET /System/Picture.
7. Mapear: para cada evento recebido, registre: endpoint (Notification/), AlarmType, campos especificos, e correlacao com o analitico configurado.

Campos chave para analitica/telemetria

- RelatedID: ID global do evento na camera (bom para deduplicacao).
- SourceID/AlarmSrcID: canal/origem.
- AlarmType: roteamento para o analitico.
- TimeStamp: ordenar/gerar linha do tempo.
- AlarmPicture/ImageList: vincular evidencias (imagens) ao evento.

7. Checklist de implementacao em Go

- Cliente Digest: suporte completo a nonce/realm/qop e reenvio automatico.
- Modelos de request/response: structs para Subscription (create/keepalive/delete) e parsing generico de Notification.
- Keepalive scheduler: goroutine que renova a cada Duration/2 e reinicia em caso de falha (com limites).
- Servidor HTTP de eventos: timeouts, limites de tamanho de body, logging estruturado, ACK consistente.
- Armazenamento: persistir eventos crus + versao de parser (para evoluir sem perder dados).
- Observabilidade: metricas de taxa de eventos, erros de ACK, latencia, keepalive OK/fail, e alarmes.

Apendice: calculo do Type

Type = soma(2^bit) para cada categoria desejada. Ex.: Smart alarm (bit3) -> 2^3 = 8. People count (bit14) -> 16384. Todos os bits listados na tabela (0,1,2,3,4,5,6,8,10,11,12,13,14,16) -> 97663.

Apendice: endpoints de exemplo

- POST /LAPI/V1.0/System/Event/Subscription
- PUT /LAPI/V1.0/System/Event/Subscription/<ID>
- DELETE /LAPI/V1.0/System/Event/Subscription/<ID>
- POST /LAPI/V1.0/System/Event/Notification/Alarm
- POST /LAPI/V1.0/System/Event/Notification/HatDetection
- POST /LAPI/V1.0/System/Event/Notification/PersonInfo
- POST /LAPI/V1.0/System/Event/Notification/Loitering