KRONYX Data Protection Impact Assessment (DPIA) Summary v1.0

Platform: KRONYX (Patent-Pending IoT Telemetry Engine)
Processor: Kenneth J. Hibberd, NMESYS

1. Description of Processing

KRONYX is a PAAS product of NMESYS. It processes sensor and gateway data for large-scale IoT deployments. The platform receives messages via FTP, MQTT, HTTP, and other protocols, updates live device state in memory, and archives normalized data securely for reporting and compliance.

2. Nature of Data

- Device telemetry data (temperature, pressure, flow, consumption, etc.)
- Metadata linked to building units and devices
- Pseudonymized user reference identifiers (e.g., tenant ID)
- No direct personal identifiers processed

3. Purpose of Processing

- To provide lawful and secure telemetry processing
- To support customer billing, technical monitoring, and regulatory reporting

4. Legal Basis

Processing is conducted on behalf of the controller (Customer) under:

- **GDPR Art. 6(1)(b)**: Performance of a contract
- **GDPR Art. 28**: Processor responsibilities

5. Risk Assessment

Risk Area	Risk Level	Mitigation
Unauthorized access	Low	Role-based access control, logging, encryption
Data breach	Low	Encrypted at rest (AES-256) and in transit (TLS 1.3)
Re-identification	Very Low	Pseudonymization and architectural separation of PII
Cross-border transfer	None	All data processed and stored exclusively within EU

6. Security Measures

- End-to-end encryption (TLS 1.3, AES-256)
- In-memory real-time state engine
- Sharded, encrypted archival database

- Strict separation of roles, no shared root accounts
- Daily access audits and key rotation protocols
- No access by non-EU entities or subprocessors
- No 3rd Party Access
- No sale of data

7. Data Minimization & Storage Limitation

- Only necessary metadata is retained
- Full messages are archived under retention policy with encryption & compression
- Cold storage logs available for 36-month recovery and auditing

8. Processor Rights & Transparency

- Full controller access to reports, data, and audit logs
- DPIA can be shared with DPA upon request
- Subject rights honored as directed by controller (access, rectification, deletion)

9. Evidence of Secure Data Storage (Visual Samples Available)

Screenshots of KRONYX's database schema are available upon request and demonstrate:

- Use of pseudonymized identifiers for all user and device data
- Encryption of sensitive fields
- Absence of direct personal identifiers
- Strict separation of access by function and role
- Example sensitive data at rest

USER:


```
{
  "_id": {
    "$oid": "685ea53cccfd75115dd4f2f1"
  },
  "created_at": 1751033152750,
  "updated_at": 1751033152750,
  "user_id": "tpx4ymcevxcnf",
  "account_id": "402weo18m1t7lpqq",
  "allow_passwordless_login": true,
  "roles": [
    "accountOwner",
    "authenticatedUser"
```

```
  ],
  "scopes": [],
  "active": true,
  "__v": 0
}
```


CONTACT:


```
{
  "_id": {
    "$oid": "685ea53cccfd75115dd4f2f1"
  },
  "created_at": 1751033148749,
  "updated_at": 1751033148749,
  "contact_id": "tpx4ymcevxcnf",
  "first_name_hash":
"6a1e9e2d9b75e944ae0061145fde801b7fa05707bdbca2cba9224613747d0fd873b3133d1c
bff5f2951bd57e1ff4289a",
  "last_name_hash":
"2389e4ccfe1085ccff33e9be343a4d8c439cdcf9240d3601c86e9971c23adeb31eab95ed09
aefee8c6dd0b99460cec6f",
  "attributes": [
    "user"
  ],
  "first_name":
"08b6fa54064e7a66760594be94b2c97c170aa93dcce074b87d34cebe61f9fcf975a76098e9
3b6487b1dd936b917114ccf6aca79369a476:545a03dee0ddeb80cf6de14d5eb49b6b3544dc
6d971cda5e78c8cd81df72a22febd4dc017d1dcc6848631729e89fbd778d9b324a1e61d4bd9
b253230f650ef786e8488466d6d74eed18693988cefaf1a3f5b63a573a244a756617f2adfba
8a575fa26d3ac433fa38beb99397c1c001f8|k1",
  "last_name":
"8129c94f960ad3648096bc5fcee2bfe006d0eaf07f71d0b5bbc41ec88676316f8183519b68
96539f199f5f9e047922049d526091f16232:1b679891f40aad601f9fa7e1eb7a4cdcff6341
13e73e41eda6f9e59b8e16a4c4d37c55005c26d9e55653afc05f53e93de34c085b5147a3c7d
8991743ed0b4384a32bad4dc97e3ae3f3c02035f820f1742a3acc4f7e894110dca73d43383d
b44810c00ca7f605511fb58216d0d7294b0d|k1",
  "address_ids": [
    "m3od56u4"
  ],
  "phones": [
    {
      "type": "work",
      "number":
"73eafccd38f268480c19bba88adbc59080f1c4c02987f753069465e14d143997a2641286ee
accc4bf9248156569ecca8261471b70b182f1c6871c2d6:0187d798a7b5f00a392a3c33a804
884d141f11aee0556b4afe1a717be8904856aed88e617cad5aaf26bc8d0a9112371bceccf75
bcf346575e8be4e58f69e0b3f4e8a63b76607f812208ce34ecdaaca384024a9c375df30af75
edfde1183d0f42e25235081d71881636569b451cc98412|k1",
      "hash":
"dc4267b66dc709fefc6aa8d63d06aabb89bcd6bce907a5b51f80ef471abee800dbe2b054e6
2bb10fae9fe5664327f9b6"
    }
  ],
```

```
  "emails": [
    {
      "type": "work",
      "address":
"01bea71cb7b4ef13c573f0bd815b17e156b5489da2c21bcdbdcd93610ce40f7099a6fbef4c
7402c446c706b348347df28c5380edab6db12ba66616c3ccfc0533e0b1:ece35b00d12f3b96
e243da9bc2dd8c51b17d0c831224695d57899e53ca8bf3db13be11e50e13feb41d198f8a999
c9a70ea1b9da3b7879f8eef8357904b2aa30d118b4f872822a2bc8568c858c5a9bf11a5913c
9ca1035f93ee0bb3986434a4f5be926b895bad88f9b53a85c64e01c271|k1",
      "hash":
"39973b4e06eebd64c3d79a01445b51d78ebd69707d032730e77ee4ffee41cd0d8914ced3fd
22a4dce8c2609bff898bd8",
      "verified_at": 1751274679264,
      "gravatar":
"d13da65d9561d8c7b37063c2da3523ece160af1949bc1c0a80a9d5cc527ba7e0efabc653d9
1c85293f1331902cc9a5c4a870e0d360080e0a4789c5e042328715f568e61f538c11026cfaf
ec39b10229bf947e1b4c67ff1d64b5f6033c23b8eae72c38a8862ea95d138df1ec13d08abb3
7d0c28aca4ebe64fc3bc699811ba3a6e78a7164696a21678399f493cbb70dad8e75d0c305a2
1350451b026ce39793f87224c:620e1e1b0a57f63a22b3457dda7f6e990a2f158f91135b529
26d318b9ae7a07f346d9883e30a1f4d2c8c10a711e2b0bb4f143eab2b2bc29abfceec7ae01e
c2b01031b4703d873c98d83355edfccc21efe0d5074613026d36cf818c00556dfa03616c50e
a4605398abbc2378d3f806cfe|k1"
    }
  ],
  "tags": [],
  "avatar": "",
  "eSocial_media":
"8eb861b83333348c4f019f97f238eea0dd723b265f8fcde6b7fec1ddf6200ad9f85878b81f
fa0f735c01a1aa47d4aac772484de9c44d4531963e706ffaa62e5f92c55712b1b40ce8a4f76
131650a634db73d424d33d4e1812b93fff1f3795425936d01aca9277efae08261236683ee34
1bb068fd624a2db436290aedef7d3bb77e35ded4d090786656b4c90c02f695205a89a8a5ecd
08304485fd4fa1bcdca793779:fa0a5378de14e460381c44658575ad224665ba030bd624906
e2b44fd54875daf45e9a4c61371ba6c57c9e07f8eb8e6f4fba8797c9ca5386592d80bca7328
85de89829358f79615eb97b9dc29fba3a301a9c38c188593c67c77c8f5f8d5e95b5cce57ce0
9008c06590d228c860d07cdbf|k1",
  "__v": 0,
  "account_id": "402weo18m1t7lpqq"
}
```

10. Advanced Encryption & Hashing Implementation

- All plaintext fields (e.g., names, contact details) are fully encrypted using AES-256
- A unique, symmetric AES key is randomly generated per encryption operation
- This AES key is encrypted remotely using a secured master key server and returned alongside the encrypted payload
- First names, last names, and other searchable identifiers are hashed using SHA3
- Lookup operations (e.g., by email) are performed against the SHA3 hash, not the plaintext
- Passwords and API tokens are hashed using Argon2i with unique salts for each entry
- Decryption is possible only through the authorized KRONYX decryption routine, which validates origin, user permissions, and integrity before granting access

Example Search Hash:

Where Red is the encrypted email address and Blue is the SHA3 hash allowing search and query by email address.


```
"emails": [
    {
      "type": "work",
      "address":
"01bea71cb7b4ef13c573f0bd815b17e156b5489da2c21bcdbdcd93610ce40f7099a6fbef4c
7402c446c706b348347df28c5380edab6db12ba66616c3ccfc0533e0b1:ece35b00d12f3b96
e243da9bc2dd8c51b17d0c831224695d57899e53ca8bf3db13be11e50e13feb41d198f8a999
c9a70ea1b9da3b7879f8eef8357904b2aa30d118b4f872822a2bc8568c858c5a9bf11a5913c
9ca1035f93ee0bb3986434a4f5be926b895bad88f9b53a85c64e01c271|k1",
      "hash":
"39973b4e06eebd64c3d79a01445b51d78ebd69707d032730e77ee4ffee41cd0d8914ced3fd
22a4dce8c2609bff898bd8",
...
```


Example Encryption Schema:

Where Blue is the encrypted payload. Red is the encrypted random AES256 Symetric Key and Yellow is the key version ID.


```
8eb861b83333348c4f019f97f238eea0dd723b265f8fcde6b7fec1ddf6200ad9f85878b81ff
a0f735c01a1aa47d4aac772484de9c44d4531963e706ffaa62e5f92c55712b1b40ce8a4f761
31650a634db73d424d33d4e1812b93fff1f3795425936d01aca9277efae08261236683ee341
bb068fd624a2db436290aedef7d3bb77e35ded4d090786656b4c90c02f695205a89a8a5ecd0
8304485fd4fa1bcdca793779:fa0a5378de14e460381c44658575ad224665ba030bd624906e
2b44fd54875daf45e9a4c61371ba6c57c9e07f8eb8e6f4fba8797c9ca5386592d80bca73288
5de89829358f79615eb97b9dc29fba3a301a9c38c188593c67c77c8f5f8d5e95b5cce57ce09
008c06590d228c860d07cdbf|k1
```

11. Roles & Scopes based Router Resource Control

KRONYX uses Fastify-based scoped routing logic to restrict access to data based on assigned user roles and scopes. All client requests are authenticated and inspected through secure session cookies signed by the IronToken system. These requests inject user metadata into the `req` object, ensuring that access is both user-specific and traceable.

Below is a simplified example of how access is enforced per endpoint:

```
function checkPermissions({ req, requiredScopes = [], mode = 'any' }) {
  const { user } = req;
  if (!user?.user_id || !Array.isArray(user.scopes)) throw new
Error("Unauthorized")
  if (user.roles?.includes('god')) return true;
  const userScopes = new Set(user.scopes);
  const authorized =
    mode === 'all'
      ? requiredScopes.every(scope => userScopes.has(scope))
      : requiredScopes.some(scope => userScopes.has(scope));
  if (!authorized) throw new Error("Unauthorized")
  return true;
}
```

This is paired with request handlers that validate both scopes and ownership via `account_id`:

```
fastify.get("/", {
  preHandler: async (req, reply) => {
    try {
      checkPermissions({ req, requiredScopes: ["account.view.own"] });
      requireAccountMatch(req);
      await Validate.async(req.query, {
        account_id: RULES.stringFieldRuleRequired(),
      });
    } catch (err) {
      if (err.message.includes('Unauthorized')) {
        return reply.status(403).send({ error: "You are unauthorized to
make this request." });
      }
      return reply.status(400).send(err);
    }
  }
}, async (req, reply) => {
  try {
    const cacheKey = CACHESERVICE.keygen({
      namespace: CACHE_NAMESPACE,
      route: "/account",
      user_id: req.user.user_id,
      params: req.query
    });
```

```
    let refresh = false;
    if(req.query.refresh) refresh = true;

    if(!refresh){
      const data = await CACHESERVICE.get({ key: cacheKey });
      if(data) return reply.status(200).send(data);
    }

    const account = await AccountModel.get(req.query.account_id, refresh);
    if (!account) return reply.status(404).send("Account not found.");

    await CACHESERVICE.set({ key: cacheKey, data: account });
    return reply.status(200).send(account);

  } catch (err) {
    console.error(err);
    return reply.status(400).send(err);
  }
});
```

This design enforces strict role- and scope-based access control, logs access contextually, and ensures that each query is tied to the authenticated user's `account_id`. As such, data exposure is structurally impossible across unrelated tenants or accounts.

12. IronToken-Based Client Authentication System

KRONYX employs a hardened authentication mechanism built on the IronToken framework (NMESYS) to ensure that only verified, session-bound users can access protected resources. The process includes the following steps:

1. **Client Initialization**:
    - A random `APP_KEY` is generated client-side and stored in a Secure, Signed Cookie.
    - This key is sent with every request in the header (`x-appkey`) to tie the session to a specific device.
2. **User Login**:
    - Email address is hashed and used to look up the user securely.
    - Passwords are verified using **Argon2i** hash comparison.
    - Password are optional. The base platform is PASSWORDLESS where a random password is generated for 5 minutes and sent to the user email address. The user has the option to set a fixed password which shall expire every 90 – 180 days.
    - Upon success, an IronToken is created using a 32-byte random token and stored as a signed cookie with a max age of 3 days.

3. **Session Payload Storage**:
    - A Redis payload is stored with the token as the key, encrypted and containing:
        - `user_id`
        - `cacheKey`
        - The original `APP_KEY`
    - Redis expiration is synchronized with the cookie expiration.
4. **Request Verification Middleware**:
    - For every API request, the IronToken cookie is unsign-verified.
    - Redis is queried for the encrypted payload and decrypted.
    - The `APP_KEY` from the client is compared to the one stored in Redis.
    - The user object is retrieved and injected into `req.user`.
    - All microservices rely on this injected identity for authorization.
5. **Proxy Gateway Enforcement**:
    - All backend APIs are accessible **only** from an internal Docker network.
    - A Fastify-based proxy performs verification and forwards authenticated requests only.
    - Critical metadata (`x-irontoken`, `x-user`, etc.) is injected into headers for downstream services.

This setup ensures that:

- Only trusted, session-bound devices can issue requests
- Session hijacking is mitigated by client-bound `APP_KEY` validation
- Tokens are short-lived and revocable via Redis session destruction
- Backend services never operate on anonymous or client-supplied tokens directly

KRONYX is designed from the ground up to exceed GDPR technical expectations for telemetry platforms. No unencrypted identifiers are stored. Access is restricted to authorized EU-based KRONYX operators only.

Prepared by: Kenneth J. Hibberd
Date: 01 July 2025

Kenneth J. Hibberd
Chief Architect, NMESYS

email: kenneth.hibberd@nmesys.io
mobile: 0160 9577 6948
office: 09723 936 9085