



Vertex

Synapse Bootcamp

Module 7

Pre-Storm Background

v0.4 - May 2024



Objectives

- Compare the Synapse UI and Storm
- Review the Synapse data model
- Explain the data model namespace
- Understand how knowing the data model relates to using Storm
- Know some advantages of the Storm query language



The Synapse UI and Storm

- So far we have focused on the Synapse UI
 - Make it easy to start using Synapse
 - Help analysts be productive from day one
- In many cases the UI simply runs Storm for you
 - Simplify common analyst tasks
 - Synapse runs on Storm!
- Using Storm directly gives us more flexibility
 - Do anything - not just what is exposed through menus and icons

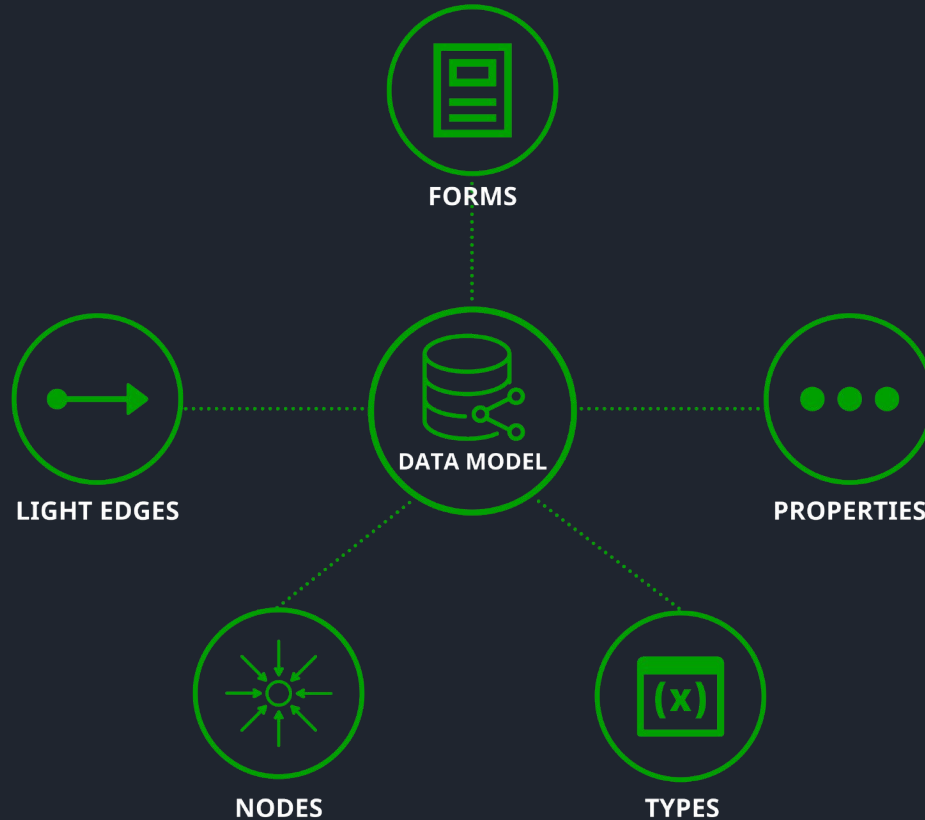
Storm **complements** the UI and provides an additional, powerful option.



Data Model



Data Model Elements





Forms, Properties, and Nodes



Forms

- "Templates" for creating nodes
 - Consists of **properties** and **types**
- Form name = primary property
 - Secondary properties
 - Universal properties
 - Extended properties

IPv4 `inet:ipv4` [Diff in Research Tool](#) [docs link](#)

An IPv4 address. type: `inet:ipv4`
example: 1.2.3.4

Properties

name	ro	type	doc
<code>:asn</code>		<code>inet:asn</code>	The ASN to which the IPv4 address is currently assigned.
<code>:dns:rev</code>		<code>inet:fqdn</code>	The most current DNS reverse lookup for the IPv4.
<code>:latlong</code>		<code>geo:latlong</code>	The best known latitude/longitude for the node.
<code>:loc</code>		<code>loc</code>	The geo-political location string for the IPv4.
<code>:place</code>		<code>geo:place</code>	The geo:place associated with the latlong property.
<code>:type</code>		<code>str</code>	The type of IP address (e.g., private, multicast, etc.).
<code>.created</code>	—	<code>time</code>	The time the node was created in the cortex.
<code>.seen</code>		<code>ival</code>	The time interval for first/last observation of the node.

Extended Properties

name	ro	type	doc
<code>:_virustotal:reputation</code>		<code>int</code>	The VirusTotal reputation score.
<code>:_virustotal:votes:harmless</code>		<code>int</code>	The number of harmless votes from the VirusTotal community.
<code>:_virustotal:votes:malicious</code>		<code>int</code>	The number of malicious votes from the VirusTotal community.

A **form** shows us *how* we can create something. A **node** is the *object we create* based on the form.



Forms as Objects

- Forms can represent "**things**" in the real world (including abstract things)

Object	Synapse Form
An IP address	<code>inet:ipv4</code> or <code>inet:ipv6</code>
An article, paper, report, or document	<code>media:news</code>
A server certificate or code-signing certificate	<code>crypto:x509:cert</code>
A computer or device	<code>it:host</code>
An organized group of individuals	<code>ou:org</code>
A commercial industry	<code>ou:industry</code>
A set of contact information	<code>ps:contact</code>



Forms as Relationships

- Forms can represent **relationships** between objects:

Object	Synapse Form
A DNS A record (FQDN and the IP it resolves to)	<code>inet:dns:a</code>
A certificate associated with a host (IP / port)	<code>inet:tls:servercert</code> / <code>inet:tls:clientcert</code> (new) <code>inet:ssl:cert</code> (old)
A file signed with a code signing certificate	<code>crypto:x509:signedfile</code>
A URL hosting a file	<code>inet:urlfile</code>
A file existing in a particular location	<code>file:filepath</code>
An email message with an attached file	<code>inet:email:message:attachment</code>
A person attending a conference	<code>ou:attendee</code>



Forms as Events

- Forms can represent **events** or **instances** of things
 - Typically include a `:time` property

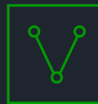
Object	Synapse Form
A specific DNS query and / or response	<code>inet:dns:request, inet:dns:answer</code>
A specific HTTP query	<code>inet:http:request</code>
A file / process interacting with a file system	<code>it:exec:file:read, it:exec:file:add, it:exec:file:write, it:exec:file:del</code>
A conference	<code>ou:conference</code>
An alert	<code>risk:alert</code>
An attack	<code>risk:attack</code>



Forms and Primary Properties

- A form's primary property must be **unique** for that form
 - o Helps ensure Synapse does not store multiple copies of a piece of data

Primary Property Category	Description	Example
Simple	Single value	<code>inet:fqdn = vertex.link</code> <code>inet:ipv4 = 1.2.3.4</code>
Composite (comp)	Two values that make a unique pair	<code>inet:dns:a = (vertex.link, 1.2.3.4)</code>
Guid (globally unique identifier)	128-bit value	<code>media:news = 6b37b042c41ea0b14aa62a42e7a4a699</code>



Why Guids?

- When it's impossible to "uniquely" describe an object with a set of values
 - What makes a "DNS request" unique?
- When the amount of data you have about something may vary

`inet:dns:request` [Lift in Research Tool](#) [docs.link](#)

A single instance of a DNS resolver request and optional reply info. type: `inet:dns:request`
base: `guid`

Properties

<u>name</u>	<u>ro</u>	<u>type</u>	<u>doc</u>
<code>:exe</code>		<code>file:bytes</code>	The file containing the code that attempted the DNS lookup.
<code>:host</code>		<code>it:host</code>	The host that attempted the DNS lookup.
<code>:proc</code>		<code>it:exec:proc</code>	The process that attempted the DNS lookup.
<code>:query</code>		<code>inet:dns:query</code>	A DNS query unique to a given client.
<code>:query:name</code>		<code>inet:dns:name</code>	A DNS query name string. Likely an FQDN but not always.
<code>:query:name:fqdn</code>		<code>inet:fqdn</code>	A Fully Qualified Domain Name (FQDN).
<code>:query:name:ipv4</code>		<code>inet:ipv4</code>	An IPv4 address.
<code>:query:name:ipv6</code>		<code>inet:ipv6</code>	An IPv6 address.
<code>:query:type</code>		<code>int</code>	The base 64 bit signed integer type.
<code>:reply:code</code>		<code>int</code>	The DNS server response code.
<code>:server</code>		<code>inet:server</code>	A network server address.
<code>:time</code>		<code>time</code>	A date/time value.
<code>.created</code>	—	<code>time</code>	The time the node was created in the cortex.
<code>.seen</code>		<code>ival</code>	The time interval for first/last observation of the node.



Forms vs. Nodes

Form (inet:ipv4)

Form / Property	Type
inet:ipv4	inet:ipv4
inet:ipv4:seen	ival
inet:ipv4:created	time
inet:ipv4:asn	inet:asn
inet:ipv4:dns:rev	inet:fqdn
inet:ipv4:latlong	geo:latlong
inet:ipv4:loc	loc
inet:ipv4:place	geo:place
inet:ipv4:type	str

Node (inet:ipv4=60.248.52.95)

Form / Property	Value
inet:ipv4	60.248.52.95
inet:ipv4:seen	(2018/05/18 09:48:15.000, 2018/05/18 09:48:15.001)
inet:ipv4:created	2021/08/18 20:51:47.651
inet:ipv4:asn	3462
inet:ipv4:dns:rev	60-248-52-95.hinet-ip.hinet.net
inet:ipv4:latlong	23.7113,120.3897
inet:ipv4:loc	tw.yun.yunlin
inet:ipv4:place	04f520ec542de2ee87d10bb083ac081c
inet:ipv4:type	unicast



The Data Model and Storm



Why All This Data Model Stuff?

- Storm allows you to "ask questions" of your Synapse data
- The data model is the "language" you use to tell Storm what to do

Question	Storm Example
"What do we know about the FQDN evil.com?"	<code>inet:fqdn=evil.com</code>
"What IP addresses has evil.com resolved to?"	<code>inet:fqdn=evil.com -> inet:dns:a -> inet:ipv4</code>
"What files communicate with evil.com?"	<code>inet:fqdn=evil.com -> inet:dns:request -> file:bytes</code>



Storm "Vocabulary"

Object	Example
Form	<code>inet:ipv4</code>
Form and value	<code>inet:ipv4 = 217.174.156.100</code>
Form and property	<code>inet:ipv4:asn</code>
Form and property and value	<code>inet:ipv4:asn = 9009</code>
Tag (as a node)	<code>syn:tag = rep.eset.sednit</code>
Tag (as a label on a node)	<code>#rep.eset.sednit</code>

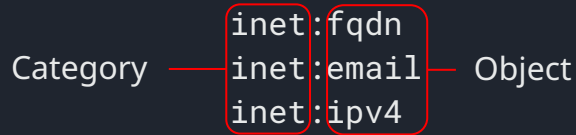
Let's look more closely at how we name forms/properties (and why)!



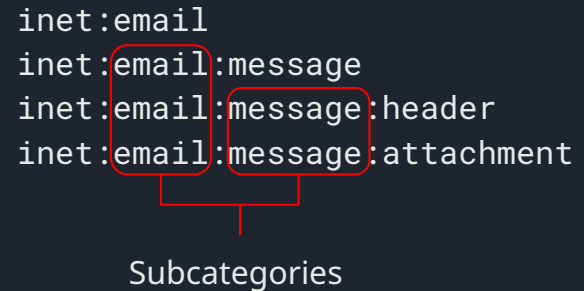
Form Namespace

- At least two elements separated by a colon (:)
 - o Category: broadly groups related forms
 - o Name: the form (object) within that namespace

Simple namespace:



May include subcategories:





Form Wildcards

- Use the wildcard (*) to match one or more namespace elements

- Instead of:

```
hash:md5 hash:sha1 hash:sha256 hash:sha512
```

- Can use:

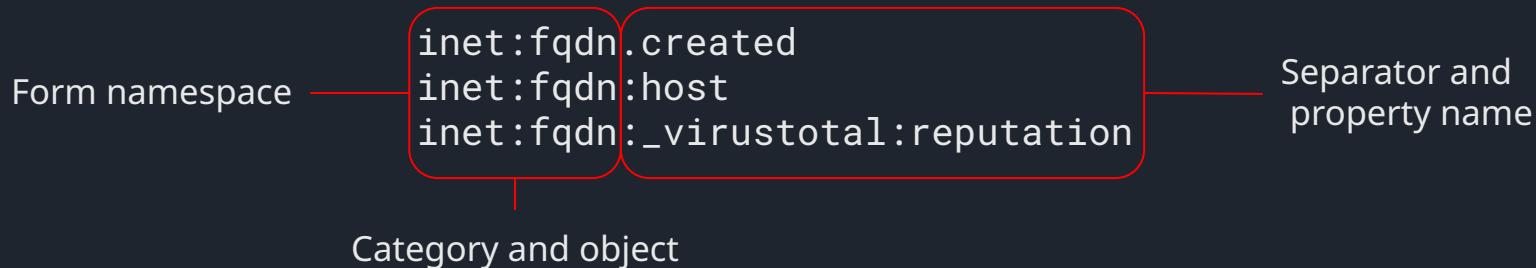
```
hash:*
```

Tip: The wildcard is a specialized use case, but handy when you need it!



Property Namespace

- Properties exist within the form's namespace
 - o Different **separator** depending on the kind of property



- Properties may have their own hierarchies:

```
file:bytes
file:bytes:md5
file:bytes:mime
file:bytes:mime:pe:compiled
file:bytes:mime:pe:imphash
```



Properties: Full vs Relative

- **Full property name:** form and property names together
 - o `file:bytes.created`
 - o `inet:fqdn:host`
 - o `inet:dns:request:query:name:ipv4`
 - o `inet:ipv4:_virustotal:reputation`
- **Relative property name:** the property name alone
 - o `.created`
 - o `:host`
 - o `:query:name:ipv4`
 - o `:_virustotal:reputation`

Understanding form and property naming conventions will make it much easier to use Storm!



Types



Types

- Every property has a **type**
 - o What kind of data does Synapse expect?
 - o `type = inet:fqdn`
 - A value that looks reasonably like a domain / FQDN
- Types provide three key advantages:
 - o **Type enforcement:** how Synapse helps keep data **tidy**
 - o **Type-specific behavior:** how Synapse helps you **work with** data
 - o **Type awareness:** how Synapse helps you **navigate** the data

Many data storage and analysis systems only use a very simple set of types - for example, integers, Booleans, and strings.

Form / Property	Type
<code>inet:ipv4</code>	<code>inet:ipv4</code>
<code>inet:ipv4.seen</code>	<code>ival</code>
<code>inet:ipv4.created</code>	<code>time</code>
<code>inet:ipv4:asn</code>	<code>inet:asn</code>
<code>inet:ipv4:dns:rev</code>	<code>inet:fqdn</code>
<code>inet:ipv4:latlong</code>	<code>geo:latlong</code>
<code>inet:ipv4:loc</code>	<code>loc</code>
<code>inet:ipv4:place</code>	<code>geo:place</code>
<code>inet:ipv4:type</code>	<code>str</code>



Type Enforcement

- Defines how data should "look" in Synapse
- In many systems, a URL is simply a **string**
 - o Potentially allows creation / input of invalid URLs

```
fd8q3:523/pogostick&?derp=hahahaha.net
```

```
visi@vertex.link
```

- In Synapse, a URL is an **inet:url**
 - o Must include a protocol header - string followed by **://**
 - o Must include a valid hostname (FQDN or IP address)
 - o Prevent a lot of bad data (though not all)

Protip: If you see a `BadTypeValue` error, something tried to set an invalid property *value*, given the property's *type*.



Type-Specific Behavior

- Custom optimizations for some data types
- An IPv4 address (`inet:ipv4`) is a 32-bit binary number
 - o 01111011011110000110111010010000
- Synapse **stores** IPv4 addresses as decimal integers
 - o 2071490192
- Synapse **shows us** (represents) an IPv4 as dotted-decimal
 - o 123.120.110.144
- Synapse allows us to **enter** an IPv4 address as:
 - o Dotted decimal, decimal integer, hexadecimal, IP range, CIDR block...

Protip: Many optimizations "just work". Some make things easier when using Storm. See the Storm documentation for details.



Type Awareness

- Synapse is "aware" of the **type** of each property
 - o Allows Synapse to "understand" and **navigate** the data model
- Most objects (nodes) are not explicitly linked
- If I ask about a specific object:
 - o `inet:ipv4 = 27.102.106.149`
- Synapse can show me all the other nodes where that IP is a **property**:
 - o DNS A records (`inet:dns:a`)
 - o Network flows (`inet:flow`)
 - o URLs (`inet:url`)
 - o SSL / TLS certificates (`crypto:x509:cert`)

Protip: Type awareness is a big part of how "Explore" in Synapse (or pivoting in Storm) works!



Type Awareness

- Nodes with **properties** with the same **type** and **value** are **connected**

Form / Property	Value
inet:fqdn	blog.cdsend.xyz
:domain	cdsend.xyz
:host	blog
:issuffix	False
:iszone	False
:zone	cdsend.xyz

Form / Property	Value
inet:dns:a	(blog.cdsend.xyz, 91.195.240.12)
:fqdn	blog.cdsend.xyz
:ipv4	91.195.240.12

Form / Property	Value
inet:ipv4	91.195.240.12
:asn	47846
:loc	de
:type	unicast



Protip: Type awareness is how "Explore" in Synapse (or pivoting in Storm) works!



Light Edges



Light Edges

- Used to **connect** nodes without a property-based relationship
 - A public report (`media:news`) that contains IOCs (such as `hash:md5`)
 - A source (`meta:source`) that provided data (such as `file:bytes`)
- Light edges have:
 - A **name** ("verb")
 - Describes the edge (e.g., "refs" for "references")
 - Distinguish different types of relationships
 - A **direction** (like an arrow)
 - The relationship only makes sense one way
 -  "The article references the MD5 hash"
 -  "The MD5 hash references the article"

```
media:news:publisher:name=eset -(refs)> inet:fqdn=evil.com
```



The Synapse UI and Storm

- What does Storm unlock for us?

Synapse UI	Storm
Query bar: Lookup / Text Search modes	Query bar: Storm mode
Easily work with common indicators	Work with any element of the data model
Easy, broad-based navigation - discover / explore	Narrowly target your research - precisely answer specific questions
Quick access to common Power-Ups / commands	Customize execution when needed
No automation	Leverage for automation!

Protip: Most analysts use a **combination** of UI elements and Storm, based on need and preference!



Summary

- The key components of Synapse's data model are **forms, properties, nodes, types,** and **light edges**
- Types support **type enforcement, type-specific behavior,** and **type awareness**
- Forms and properties use a **structured namespace** with most elements separated by a colon (:)
- The data model is a large part of the vocabulary used with **Storm**
- Using **Storm** complements the Synapse UI and gives you additional power and flexibility