Connecting to the EHR with Application Programming Interfaces (APIs)

A guide for Innovators at UCSF and beyond

June 2017

Prepared by the UCSF Digital Diagnostics & Therapeutics (DD&T) Committee

Contents

Who can benefit from EHR API access?

Have you ever thought "I could write an application that provides personalized clinical advice… if only I could get three pieces of data from our Electronic Health Record (EHR) about a patient—say, her age, weight, and creatinine level?" EHRs like the one at UCSF have gotten better at making data like this available to facilitate app creation. We want to help you tap into these possibilities so you can build innovative digital solutions as quickly and easily as possible.

There are many types of people who can benefit from learning how to access UCSF's APIs:

- A clinical researcher developing a decision-support application for physicians
- A UCSF faculty doing a QI project creating a custom dashboard for your clinic
- A trainee creating a clinical calculator
- A UCSF faculty looking to solve a clinical problem by working with a commercial software company and looking to integrate the tool with our EHR

What are APIs?

Application Programming Interfaces (APIs) are a way to move data out of or into a given program using a standard programming language. For example, if you want to know a patient's date of birth in a given EHR, you could find it manually by using the MRN to retrieve the patient record and then looking through the chart for the date of birth, or you could find it by using a software program that directly retrieves the date of birth from the EHR. The problem with the former is it is slow and not useful if you want to keep two programs in sync, and the problem with the latter is that you need to know a lot about the EHR and how it is structured (e.g. relational vs. hierarchical; which indices exist; and how are the tables laid out and named). If instead the EHR published an API, that API can be called by any software program without needing detailed knowledge of the underlying data structure and without the developer needing help from the EHR team to set up a custom "data dump" or other lengthy process.

Imagine developing a custom diabetes dashboard that includes last A1c value, date of last retinal exam, and last note from Endocrinology pulled straight from the EHR, and which could even be combined with BG meter values — and you get a feel for how powerful such data integrations can be.

What API's are Not

The EHR API's discussed in this document are useful for interacting with specific records in Apex in real time. Typically, it's used for small sets of data, such as a patient's latest lab results

or a physician's current patient list. It is not particularly useful for retrieving large sets of data, e.g. all patients seen at UCSF Cardiology in the last 10 years.

If you do not need the most up-to-date data and/or need large amounts of EHR data, consider alternative data sources, such as the Epic Clarity database, Caboodle (Epic data warehouse, formerly known as Cogito). Please refer to data.ucsf.edu for these data sources.

The EHR API's are for data exchange on demand versus being event driven. For example, if you need to know when a patient has been admitted, an API isn't useful for determining that (You'd have to poll the EHR and ask it every few minutes if the patient has been admitted. It's doable, but not practical). Instead, you'll want to use a traditional HL7 interface, e.g. the ADT interface, which sends outbound messages to subscribed recipients regarding which patients have been admitted, discharged, or transferred. Once the message that a patient has been admitted has been received, an API could then be used to query further details about that patient. For more information on HL7 interfaces, please contact the Clinical Systems Integration Team.

Sounds great! How do I get this data using API's?

Glad you're interested! The process is described in greater detail below.

While APIs can be very powerful, there are several potential risks involved, including:

- Exposing protected health information (PHI)
- APIs breaking without warning
- External applications not working as intended
- External applications slowing down the EHR for everybody

Typically, a developer wanting their app to connect to UCSF's EHR APIs would have to obtain approvals from many different stakeholders across UCSF, a process we know can be lengthy, confusing, and inefficient.

The DD&T consolidates this approval process and guides you through to save you time and confusion.

After reading through this guidance, when you are ready to request API access, please complete this intake form. A PDF version of the intake form is available here for you to preview.

Ready to begin? Great—keep reading.

DD&T Process Overview

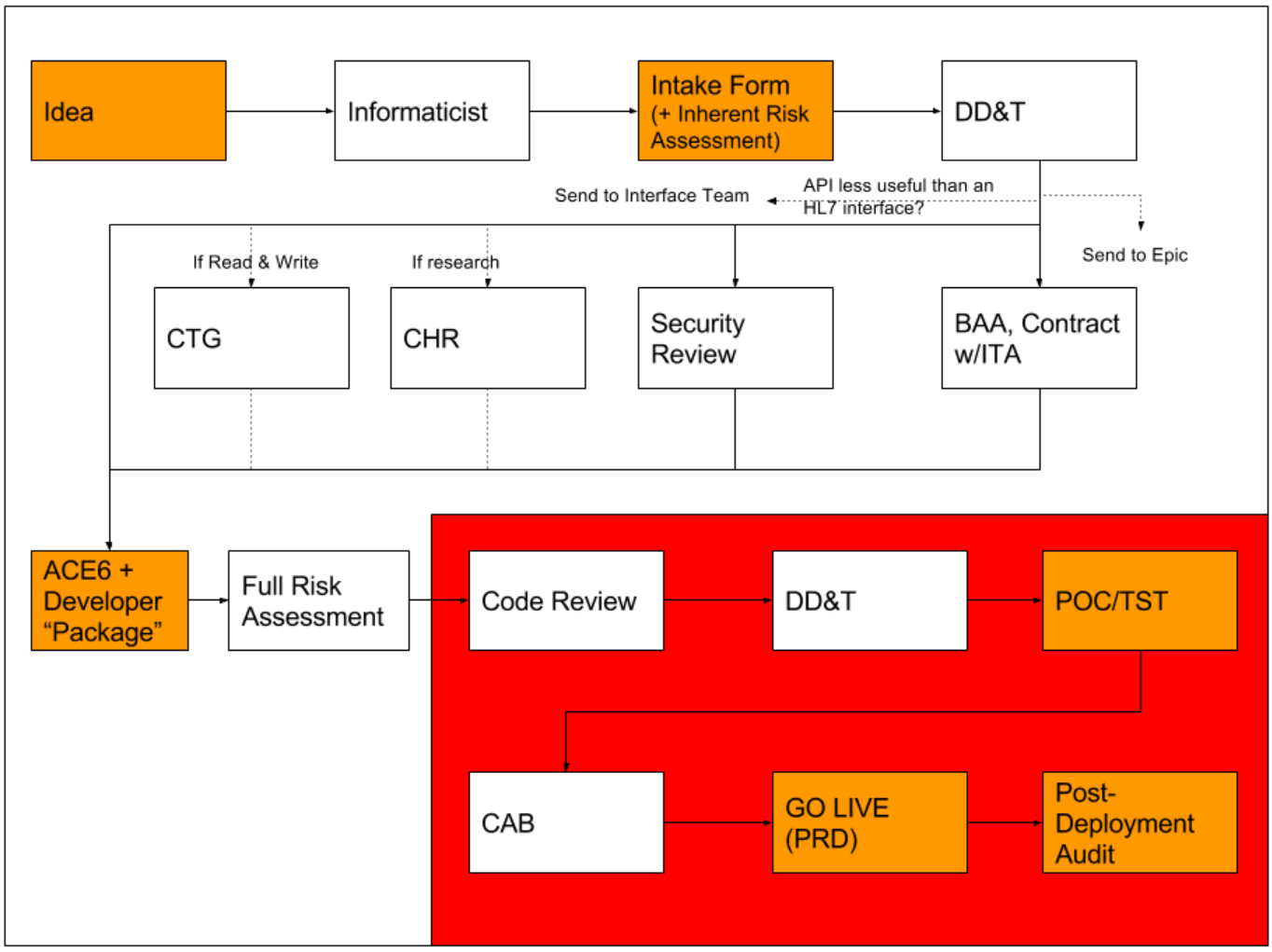

Figure 1 is a map of the process behind DD&T and connectivity to UCSF's EHR APIs. These steps are explained below.

(https://docs.google.com/drawings/d/15qOrHc3fDma90HxX8GbJIw_Vi9ckv50uhtdG2xQuJB0/edit?ts=591b4875 is a link to a new version in progress)

Part 1: Moving From an Idea Into EHR Sandbox Testing

Step 1 - Develop an idea

At the very beginning, you will have an idea for an application that gets data from or pushes data to the EHR —better medication reconciliation; a diabetes dashboard; etc. You will articulate answers to questions such as, "What application is being built?", "What is the clinical need?", and "what keeps me from doing this in the EHR?". If you're not faculty in one of the standard series (Ladder, In Residence, Clinical X, HS Clinical, Adjunct) then you will need to identify a sponsor who is.

Step 2 - Pre-Review: Propose your idea to a DD&T Informaticist

Next, you'll propose your idea to a UCSF DD&T clinical informaticist (Raman Khanna, Aaron Neinstein, Priyanka Agarwal, or Andrew Auerbach). One of the physicians in the group will walk you through questions to consider and ensure that your proposal is not something that can already be done within our EHR, suggesting alternatives if these already exist.

Step 3 - Complete DD&T committee Intake Form and security risk assessment

DD&T's Intake Form helps the DD&T assess your project on various axes—scale (how many patients), scope (what does it do), and risk (what happens if it fails in some way), among others. Your application also needs to pass a high-level screening security review (for "inherent risk").

Step 4 – 1st Review by Digital Diagnostics and Therapeutics (DD&T) Committee

Your request will be reviewed by the DD&T Committee after you complete the Intake Form and Security Assessment. If your request is approved, you will be able to move forward with connecting to EHR APIs to "sandbox testing" of your app in the "ACE6" environment.

Included in this 1st DD&T Review are approvals from:

- UCSF Privacy Office
- Legal
- Risk
- Compliance
- UCSF ITA (Industry, Technology, & Alliances)

What criteria does DD&T Committee use during its 1st Review?

1. Is the external application functionality already performed by the EHR itself? If so, what is the value added by this application?
2. Does the application respect core principles of the UCSF EHR, such as retaining 'single source of truth?'
3. Do the application's clinical recommendations align with other UCSF Health standards (e.g. medications on UCSF formulary)?
4. Does the use of the API align with other key UCSF Health pillars, as articulated in the UCSF True North strategic plan?
5. Do the APIs you want to use already exist?
6. Is your app's connection likely to be costly to maintain (e.g. how many times does it call our EHR, for what information, and in domains that change frequently or otherwise)? What is cost of maintenance?
7. Are you or your software partners capable of assuming the risks of connecting to and handling Protected Health Information (PHI)?
    a. Are your security procedures consistent with UCSF standards?
    b. Can you secure PHI in transit and at rest?
    c. If you're using an external vendor, have they already or will they sign UCSF's Business Associates Agreement?
    d. Are your partners (if any) in good standing as University of California (UC) vendor?
    e. What assurances can you provide regarding safeguarding/auditing data?
8. Is this a read only application or a read & write?
9. If working with an outside vendor, what is the nature of the business relationship and what are UCSF's intellectual property interests in the project?
10. If working with an outside vendor, are they set up as an approved vendor in Epic's App Orchard?

Step 5 - Business Associates' Agreement (BAA) & Contract

After you get approval from the DD&T Committee to test your app, you will simultaneously:

1. Get access to the ACE6 "sandbox" testing and development environment (see below)
2. Be asked that your company partner to sign a Business Associates Agreement. The BAA confirms that the ***company*** will assume risk and indemnify UCSF for any PHI lost.
3. Be asked for the company partner to sign a contract regarding their work.

Step 6 - ACE6 Access & the "Developer Package"

At this point, you will be developing your application and you will receive the "developer package":

- Access to UCSF's EHR sandbox testing environment (ACE6) as a clinical user (e.g. you can log in)
- Access to the APIs connected to the ACE6 environment
- Access to API documentation on the Epic UserWeb

In ACE6, you can conduct testing to see if your application is working—if you change a medication or add a new vital sign value, is your application able to successfully reflect this by means of the EHR; etc. Demonstrating successful testing is necessary for gaining approvals to move into the "live" Epic production environment.

Note also that in connecting to the APIs, one may be connecting either directly to the Interconnect API, or to an API manager ("Wrapper") built on a separate platform that calls the IC API. UCSF reserves the right to change between the two, and require you to do so as well.

Part 2: Moving from "Sandbox" to the "Production Path"

Congratulations! You have moved your application beyond the planning stages, tested it, and now think it would be a great idea to be able to use it in the production environment. Let's walk next through the steps involved in this second part of the process, which entails a higher degree of scrutiny.

Why two separate steps and why are the requirements different?

EHR data can be difficult to manage and applications often fail to reach maturity for a variety of reasons—loss of interest, validation not working as intended, poor usability for reasons beyond the developer's control (see below). We also owe it to our patients to be extra careful because health applications are not like medications—there is usually no data from randomized trials to prove which ones are effective and which ones are harmful. And the consequences could be severe—imagine an application that incorrectly calculates a MELD score and thus gives a patient hope that she/he might receive a liver (or worse, actually misallocates a transplantable liver), or one that recommends insulin dose titrations ten times larger than usual due to a glitch in ordering parameters.

As such, not all applications will move from testing to production. DD&T's goal is to *reduce* the barriers to testing and validating brilliant ideas, while maintaining a strict process for getting these ideas into the EHR where they can help, or potentially harm, our patients.

Step 1 – Security Assessment – "Full Risk"

Once you have validated your code in the ACE6 testing environment, it will then be time to undergo a security assessment for "full risk." This assessment uses the same form as your initial security review ("inherent risk"), but will be re-examined given the fact that PHI will now exist in your application and will transit back and forth with our EHR.

Step 2 (as needed) - Committee on Human Research (CHR)

In certain cases, and particularly if the application is part of a research study, the developers will need to complete a CHR (IRB) application and provide the approval number.

Step 3 (as needed) - Care Technology Governance Committee (CTG)

If the application wishes to write data to EHR, is costly to implement or maintain, and/or is accompanied by capital budget requests, the UCSF Care Technology Governance Committee (CTG) will also need to review and approve your project.

Step 4 - Code Review (& Links within EHR)

Here, a developer from the Clinical Systems team—which is responsible for configuring our EHR for local use, e.g. what is in your post-operative pain order set and how your neurology note template behaves—will review the calls your application is making into the EHR. Simultaneously, one of the Informaticists may review your application user interface (UX) with you to confirm usability. The key idea here is to ensure the application is not creating dangerous subroutines that could either impair the performance of production environments or create backdoors for the stealing of data (not by you!), and to make sure that an application that you think is eminently usable is, in fact, not going to get used because the number of clicks that seem reasonable to you don't fit with our experience of EHR applications that actually get used. (See pitfalls below for famous examples.)

Launch From within EHR

If your application has reached this point, some questions that will naturally arise include:

1. How will a user get to my application?
2. When they do, will they need to log in?
3. When they log in, will the computer know which patient to bring up or will they have to search again?

Luckily, the EHR increasingly has ways to address the above issues, and the simplest solution will be a button in the EHR in a visible place (a column in your list; a button in your navigator; or a link from the patient header) where clicking it launches your application, logged in based on your EHR log in, with whichever patient you are viewing in the EHR. These button builds are popular and useful but do take time and we need to ensure that the placement of a CIWA management tool does not distract an ophthalmologist managing glaucoma. (This is why most of the time the header will not be the right location.)

Step 5 - DD&T Final Review

At this point, you have been through the initial DD&T committee review, achieved "sign off" through many UCSF regulatory and compliance steps, tested your application in ACE6, and had it undergo code review. You are now ready for your application to undergo its final DD&T review.

At the Final DD&T Review, your project's evolution, testing data, and potential impact are discussed and a plan is made to put it on the path to treating real patients in the EHR.

Part 3: Final EHR Deployment – Moving from "Sandbox" to the "Production Path"

Step 1 – Perform "preproduction testing" of your application in APeX POC and TST (EHR "proof of concept" and "test" environments)

Your application can now re-aim your APIs to the Epic POC and TST environments. While POC is similar to ACE6, code from POC does sometimes get migrated directly to the production environment, so we like to keep it as clean as possible. TST is like POC except it actually connects to external test systems—so for example you can order a basic metabolic panel in TST and it will actually create a requisition on the SUNQUEST system which a SUNQUEST technicians can "fill", to allow population of lab results into this environment through the usual process. This can also serve as a real-world check on whether data is flowing how you think it should and whether it is formatted in the ideal way.

Step 2 – Clinical Systems Change Control Board (CAB)

After you have gone through all of the above, the application will go to the Change Control meeting (CAB) for approval.

Step 3 – APeX Production Deployment

Once the code has been approved at CAB, you and your developer will point the production version of your application at the production APIs. Congratulations! You are now using your application to help real live patients.

Step 4 - Post-Deployment Review

One month after production deployment, you and your developer will submit a report regarding how much your application is being used, how often it is calling the EHR API, how it is affecting clinical behavior or outcomes, and any additional lessons learned. We reserve the right to request usage and access logs on demand, the same as our Privacy office demands of our internal systems, in the event of a suspected privacy breach or litigation.

Development Pitfalls

Here are some of the pitfalls we have learned about over the years when developing these sorts of applications.

1. **Consider the browser.** Many application developers start with the assumption that they can do all of their development and testing in Chrome. DON'T BE THAT PERSON. Test and validate in a variety of browsers. Previous projects we have worked on have either lost time or required massive accessibility concessions and workarounds when this factor was overlooked.
   Not only is Chrome difficult to work with in the medical environment for all sorts of cache-related reasons (have you ever opened a form in Chrome and noticed that the previous user's personal and financial information is auto-complete-saved to it, probably entirely by mistake? Us, too!), it is not the browser that many EHRs themselves use. This may change, but why set yourself up to fail?
   Which brings us to:
2. **Don't assume that run of the mill code runs on the EHR browser.** Even simple things like type-ahead, when they rely on javascript, have been known to cause problems. The power of being able to launch your application directly from the EHR needs to be balanced against the limitations these browsers impose. In general, the simpler/more accessible format beats the browser you need to open independently, no matter how many bells and whistles the latter may have.
3. **You must test as if you are a transitory user.** This is harder to do than to say, but if you are testing an application at your desk, there are things you will get used to that simply will not be acceptable if you are accessing the same application at the medical center. Example: previous applications we have worked on required 10-20 seconds to log into. If you log in once a day, this may seem acceptable, but it won't if you are using 5-10 computers each day with interruptions to return a page or step into a conference.
4. **Make firewall and other network requests early.** The UCSF network team is very busy. Make sure that if you need a server address white listed, you do that at the beginning of this whole process. Likewise, if you end up needing to stand up a brand new server, request it very early. This way, after you finish your testing, you are not 3-4 months behind as you await your production server to be spun up, loaded, and whitelisted.
5. **Break the Glass**

Appendix: Considerations

The ITA and Intellectual Property

Your work above, in addition to being good for patients, is fundamentally creative and will often involve the creation of intellectual property (IP). If you are a UCSF employee, your IP is owned by the UC Regents on behalf of the state of California (but under generous terms, more so than most private employers), but if you are working with a developer you are not paying this may get more complex. For this reason, the office of Innovation, Technology, and Alliances attends our monthly meetings so they can help protect you and UCSF while ensuring your developer also gets to keep their own IP. In general, code that is written by the company (that you didn't pay for on a contract basis) will be owned by them, but if you helped them develop it by your knowledge of clinical situations, then you should own a share as well, and the ITA can help you negotiate that up front. Custom APIs and buttons in the EHR built by the UCSF Clinical Systems team are owned by UCSF.

Privacy, Regulatory, and Risk Management considerations

The presence of PHI outside of the EHR creates unique issues that need to be managed appropriately and in accordance with UCOP BFB IS-3.

Privacy Considerations

UCSF patients have a right to privacy and to ensure that, at times of vulnerability, their data are not viewed inappropriately. If inappropriate access does occur, they should have confidence that responsible individuals will be appropriately disciplined, removed and, where relevant, prosecuted.

To achieve the above, your health application will need to log its users' activity on an individual level, via date/timestamps of:

1. Access to the application
2. Access to patient lists
3. Access to specific patients
4. (where relevant) access to specific data elements within the patient

Importantly, these do NOT have to be stored in the UCSF EHR, but they should be available on demand for remittance to internal auditors as well as to external agencies such as the office of civil rights (OCR) if and as needed, even if the app is eventually turned off. (If the app is turned off, you can provide a one-time dump of the log data to our committee as long as it is clear

who accessed what, e.g. use UCSF identifiers like employee id not identifiers internal to your application). This log would ideally be in the same format as EHR's ASCII user logs.

This is also the logic behind the post-go-live audit step above.

Accessing sensitive patient's data

A special consideration at UCSF is the multiple layers of additional privacy protection UCSF employs for a subset of patients—prisoners, some employees, and public personalities. Because it is quite easy to open a chart by mistake—and thus to claim to have done so, or to do so from someone else's computer—UCSF has enacted EHR functionality whereby access to this subset of patients' data requires affirmation (and a re-entry of your password) that you do indeed need access.

Your application will need to respect the rules around the same subset of patients. Luckily, our EHR flags such patients in its APIs, and our test environment has a few such patients, so you and your developer can test and validate your own sensitive patient subroutines. This is also where the button concept in the EHR becomes so attractive—if someone is accessing your application through the patient's chart in the EHR, it means they have already confirmed they need access, so your application can check this and just make note of it. From other parts of your application, you will need to consider whether a user has already accessed sensitive information and if not, show her/him a similar splash screen as the one in the EHR with similar safeguards before she/he confirms access.

Compliance/Regulatory Considerations

One area you'll need to think about also is scope of practice. Your app may make it functionally possible for any user to do something that a doctor would ordinarily do (e.g. dose heart failure medication or diagnose depression), but if that user is not actually a doctor then this functional capability may still be legally impermissible without appropriate oversight. This will be particularly important for applications needing R&W access.

Risk Management Considerations

Risk management will need to be kept apprised of all data, especially on 3 axes:

1. Does the external application constitute a portion of the legal medical record (LMR) subject to a release of information request?
2. Is your application data available and known to UCSF lawyers in can it can help defend the institution and providers against lawsuits?
3. Is the application creating expectations for example that data should have been viewed in this alternate format but was not, and now providers are liable for content they did

not know existed? This might apply for example to a pilot in which an application posts a warning that is never viewed because the provider it seeks to warn does not have access to the application.

Fees and Level of Service

UCSF will charge fees for the above process as follows.

1. Vendors must pay $5,000 for the DD&T and security reviews and the Developer Packet.
2. Vendors must pay another $5,000 for the 2nd DD&T review, code Review, movement to POC/TST, and CAB processes.
3. Vendors with commercial software will need to separately pay EHR for connecting to the APIs.
4. These fees must be paid *prior* to the DD&T review steps as noted above, eg upon submission of the request.

If you are developing the application yourself on UCSF infrastructure that you otherwise have access to (e.g. your laptop), the fees above will be waived; however, you'll need to undergo a security review especially as and when you add a HIPAA-compliant server.

The above fees grant access to the basic package listed above and include ONLY a "do it yourself" level of service. Any requests/questions that need to be answered cannot be directed to either the help desk or to Clinical Systems; instead, you'll need to separately open a ticket with these teams and purchase blocks of support hours in increments of 10 hours at a rate of $200/hour.

Glossary

DD&T = The Digital Diagnostics and Therapeutics Committee

BAA = Business Associates Agreement

ITA = the UCSF office of Innovation, Technology, and Alliances

PLR = the Privacy, Legal, and Risk Committee

CHR = Committee on Human Research

CTG = Committee on Technology Governance

ACE6 = Developer SandBox

POC/TST = Proof of Concept, Test EHR environments (production path)

CAB = Change control Board for the EHR

PRD = Production environment (EHR).

EHR = Electronic health record

UserWeb

App Orchard

IC API

DT = Developer team

PHI