1 OVERVIEW

1.1 Provide a high-level end-to-end description of your implementation.
At its core, our implementation simply says — use traditional, existing email systems for NHIN Direct. Users would have NHIN Direct email accounts just like they have regular email accounts, and would hook them up to NHIN Direct email servers just like they do with regular email servers today. They could use their traditional email clients (client or web-based), or vendors could integrate messaging directly into their EHR products just as they do today.

The new code in our implementation is nothing more than a security layer that “plugs in” to the existing email pipeline and transparently adds S/MIME signatures and encryption using certificates bound to the source and destination endpoints. Management of these certificates can be handled by a HISP on behalf of an organization, or directly by the organization itself.

1.2 Provide a diagram representing the software blocks required for a representative end-to-end system implementation.
Most code in our implementation is strictly off-the-shelf existing email software. The two key pieces of new software required are the agent “plug-in” and a configuration interface that holds and manages certificates. Different email servers and clients will require different “glue” code to insert the agent into the pipeline; this can be done most generally as a standalone proxy, or more efficiently as a true “plug-in” dependent on the specific proprietary plug-in interfaces offered by different vendors.

The four components of an email-based solution are:

1. The email server
2. A DNS server, mapping endpoints to server IP addresses and certificates
3. The NHIN Direct agent and configuration interface
4. An email client (traditional or embedded into an HER, or an automated process)

The diagrams below detail three possible implementations of an organization using this implementation. Note that all three of these can co-exist within the overall system; organizations can choose how they are most comfortable participating.

- **“Full Service” HISP or Large Organization** — Shows how all functionality can be wrapped into a service-style installation. Email clients connect to their servers over existing email protocols such as transport-secured MAPI, SMTP, POP3, IMAP4 or HTTP (for webmail).
- **Organization using existing ISP & Gateway** – Shows how an organization could use existing ISP-based email and DNS services, inserting a gateway in their organization to handle NHIN Direct agent duties.

- **Mail client plug-in with existing ISP** – Shows how a single user (for example, an individual provider) might install an NHIN Direct plug-in directly into their email client and use existing ISP-based email and DNS services.

---

```
1. HISP is trusted agent for the organization. OR
2. Entire "HISP" can be moved behind the organization’s firewall to avoid #1
3. Cert management occurs within the HISP
4. Organization uses off-the-shelf mail connectivity solutions (including embedded functionality)
```
Organization using existing ISP & Gateway

- Existing commercial ISPs can serve as routing nodes
- No PHI visibility @ ISP
- Organization manages its own configuration within the gateway
- Can work both as organization or individual (single-user)
- Organization uses off-the-shelf mail connectivity solutions (including embedded functionality)

Mail Client Plug-in with existing ISP

- Existing commercial ISPs (e.g., Hotmail, Yahoo, Hosted Exchange) can serve as routing nodes
- No PHI visibility @ ISP
- Organization manages its own configuration within the agent
- Provider uses off-the-shelf mail connectivity solutions

HISP-HISP

HISP-Source/Destination
1.3 Describe the current overall state of the implementation

We have enough of the pieces done that we can send email end-to-end implementing the full stack as required for pilot implementations.

The NHIN Direct agent has been implemented for .NET at approximately 90% completion (working end-to-end, last 10% is about edge case configuration). The Cerner-created Java implementation is working end-to-end, as is a Ruby implementation created by Arien Malec.

Glue code for Windows SMTP Server is operational that can send NHIN Direct mail end-to-end using any email client; we have been testing with Windows Live Mail.

An integration with HealthVault that allows patients to receive messages from providers directly into their accounts, including structured CCR and CCD attachments, will be operational within days.

A general purpose Windows-based desktop gateway for small providers is in the early stages.

Configuration tools have not been started, but are also not required for pilot instantiations.

1.4 Describe the working group participants and contributions

Active contributors of code to date have included Microsoft and Cerner. Secure Exchange Solutions is participating as well and brings existing operational products to the table.

While we have been disappointed that more teams have not yet jumped on board with the email approach, it is our believe that the end-user and ISP communities will respond extremely positively to it. We also hope that as we stand up end-to-end demonstrations over the next week, the power of standing on the shoulders of something as ubiquitous as email will become clear.

2 Benefits/Drawbacks

2.1 Outline why you believe this is the best approach to achieving the goals of NHIN Direct.

For a more complete discussion of why this approach is the best match for the NHIN Direct goals and charter, please see the Wiki page The Case for SMTP. Briefly here:

- Email scales from small practices to large ones, and to patients.
- It is supported across all platforms by a ton of robust, real-world proven software, both client and server, that handles all the edge cases: admin, monitoring, provisioning, backups, anti-virus scanning, redundancy, etc.
- Its store-and-forward design is resilient to network and server outages.
- It enables both unstructured and structured exchange.
- It supports endpoints that are humans, groups, or computers.
- It uses concepts that are widely understood and accepted.
The birth of NHIN Direct was the observation that providers and patients wanted to share messages with each other. Email is what our users know, and what they want – we should give it to them.

2.2 Outline any Drawbacks to this approach
The agent approach requires certificate distribution through DNS, which while supported in virtually all running DNS servers and systems, is not widely implemented today – so this is some new ground for ISPs. However, it is important to note that the agent and SMTP are not inextricably tied together. If we find a way to use TLS – the SMTP model will still have all the same advantages.

Not all messages created in this approach could be transferred over to an NHIN Exchange network, because we do not require sufficient metadata to create the proper envelope. However, there is a straightforward path for users to include sufficient metadata, and once present the SMTP model interacts very well with the existing NHIN Exchange SMTP transport.

Some might say that there is less library-level support for email/MIME operations vs. something like HTTP and XML. There is some truth to this – but only some, and this has not shown to be a barrier for any product interested in including email interaction in their interface to date.

3 Specific Artifacts

3.1 Describe the status of working code.
See section 1.3 for a detailed description of code, which can be found in the following repositories:

- nhin-d-agent (.NET agent)
- nhin-d-jagent (Java agent)
- nhin-d-smtp (glue code)

We are already sending messages end-to-end and expect to stand up rich demonstrations in time for our face-to-face meetings the week of June 6.

3.2 Describe the status of written specifications.
The required agent behavior is described in detail in pseudocode and overview diagrams (1, 2) today. The original SMTP implementation overview on the Wiki is out of date and will be updated over the next two weeks.

3.3 Describe the status of test tools and cases.
The Java agent has basic unit tests in place. Other implementations and components do not at this time.
3.4 Describe the “production readiness” of the working code – e.g., are errors and edge cases handled, are administrative and monitoring tools available, etc.

Because we rely on so much off-the-shelf code, most of the SMTP stack is completely ready to go today at full production readiness. The .NET agent is at production quality now. Existing glue code will require bulletproofing before it is ready to ship.

4 ABSTRACT MODEL

4.1 Describe how each numbered transaction is supported by the implementation.

Note in this description I detail the transactions in the “full service HISP” model as pictured in section 1.2 because it is the most natural “fit” to the Abstract Model terminology. However, it is important to note that all section 1.2 configurations support the abstract model; the NHIN Direct Agent operations simply sit in different numbered transactions in these cases.

4.1.1 Source to HISP

Configuration

The source (mail client) must be configured with the name or IP address of its HISP (mail server), and either collect credentials from the user (e.g., through a username/password dialog box) or through enterprise configuration (e.g., federating an Active Directory login).

Transaction 1.1: Source authenticates to HISP

Having identified the appropriate credentials, the source passes them to the HISP using the appropriate source edge protocol (SMTP(S), MAPI, etc.). The HISP validates these credentials and either accepts the incoming connection or rejects it with an access denied response.

Transaction 1.2: Source sends NHIN Direct Message to HISP

The source creates an appropriately-formatted message and sends it over the authenticated connection to the HISP. In the SMTP case this will consist of MAIL FROM, RCPT TO, and DATA commands, but as previously specified alternative protocols are acceptable here as well that will have different specific exchanges at this edge.

At this point the NHIN Direct Agent takes over per the following diagram, either rejecting the message or accepting it for transfer based on the sender’s certificate configuration. Note that this diagram has been simplified to work with a single recipient message, although the Agent and code supports multiple recipients as per normal Internet mail semantics.
Sending a Message

These operations can be inserted anywhere along the pipeline from the source client through the source HISP.

Transaction 1.3: Source receives message status information

Normal email semantics apply here. If a message is rejected, it is returned to the source HISP and ultimately to the source as a new “bounce” message that is authored by postmaster@rejecting.hisp. This message will include a well-known header containing a numeric code describing the error condition (list of codes TBD). Standard sign and encrypt rules apply to these bounce messages as well.

No explicit ACK is sent automatically in the case of a successful send. However, clients may implement standard “return receipt” headers to request an ACK from the receiving system. Conceivably, a HISP could be implemented to always send ACKs, but we have not made that a part of our current implementation.

4.1.2 HISP to HISP

Transaction 2.1: Source HISP and destination HISP mutually authenticate each other

Mutual authentication of HISPs as independent actors is not required in our implementation. Instead, we mutually-authenticate sending organizations/endpoints using the NHIN Direct Agent sign/encrypt operations. However, if it is deemed useful for HISPs to also mutually authenticate, we would configure this as standard SMTPS exchange using certificates from generally-accepted CAs. Note that this is only
meaningful as a way to prevent routing information from being “sniffed” over the HISP-HISP channel; all payload protection is done by the NHIN Direct Agent.

**Transaction 2.2:** Source HISP sends NHIN Direct Message to destination HISP

The source HISP sends the message to the destination HISP using standard SMTP(S) commands. Upon receipt, the destination HISP invokes the NHIN Direct Agent to decrypt and validate the inbound message, per the following diagram. Note that this diagram has also been simplified to include a single recipient address at the destination HISP, while there may be many per typical email usage. The diagram is simplified for clarity only; the NHIN Direct Agent implementation supports this situation fully.

Receiving a Message

These operations can be inserted anywhere along the pipeline from the destination HISP through the destination client.

![Diagram of message decryption and validation process]

Local Configuration

<table>
<thead>
<tr>
<th>Anchor Certs</th>
<th>Private Certs</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Public 1</td>
<td>• Private A</td>
</tr>
<tr>
<td>• Public 2</td>
<td>• Private B</td>
</tr>
<tr>
<td>• Public 3</td>
<td></td>
</tr>
</tbody>
</table>

If the message is rejected, a bounce message is returned to the source HISP as per transaction 2.3 below. Successful messages are persisted associated with the appropriate mailbox awaiting pickup by the destination client.

**Transaction 2.3:** Source HISP receives message status from destination HISP

Normal email semantics apply here. If a message is rejected, it is returned to the source HISP and ultimately to the source as a new “bounce” message that is authored by postmaster@rejecting.hisp. This message will include a well-known header containing a numeric code describing the error condition (list of codes TBD). Standard sign and encrypt rules apply to these bounce messages as well.
No explicit ACK is sent automatically in the case of a successful receive. However, clients may implement standard "return receipt" headers to request an ACK from the receiving system. Conceivably, a HISP could be implemented to always send ACKs, but we have not made that a part of our current implementation.

Of course, if specific use cases require application-level ACKs, that will be implemented using a return message sent from the destination – and would not be recognizable as any "special" message from the HISP perspective.

4.1.3 HISP to Destination

Configuration
See Source to HISP configuration in section 4.1.1.

Transaction 3.1: Destination authenticates to HISP, or HISP and Destination mutually authenticate each other

Having identified the appropriate credentials, the source passes them to the HISP using the appropriate source edge protocol (POP3(S), MAPI, etc.). The HISP validates these credentials and either accepts the incoming connection or rejects it with an access denied response.

Transaction 3.2: HISP MAY provide the Destination a list of available NHIN Direct Messages

Standard email client/server protocols are used to enumerate and manage the list of messages waiting in the destination mailbox. In POP3, this exchange is done through the LIST command, but other protocols may be used as well.

There is a great deal of richness in these existing protocols. For example, messages may be downloaded or left on the server, headers may be reviewed before downloading full messages, and with servers supporting IMAP4 and MAPI whole folder structures may be managed within the inbox itself.

All of these are supported by our implementation.

Transaction 3.3: HISP provides an NHIN Direct Message to the Destination

Standard email client/server protocols are used to fetch messages from the HISP. In POP3, this exchange is facilitated using the RETR and possibly DELE commands, but other equivalent protocols may be used as well.

Again, existing protocols offer a wealth of options here, including downloading partial messages. All of these are automatically supported by our implementation.

Transaction 3.4: Destination updates message status with the HISP

The destination can send return messages back to the source through the HISP. These messages may be structured or unstructured as appropriate for the use case.
4.2 Describe how each defined "term" in the abstract model is instantiated in the implementation.

4.2.1 Source and Destination
The “mail client,” which may be instantiated as a traditional desktop or webmail client or an EHR or PHR with embedded email functionality. Generally will use some type of “inbox” metaphor to show current messages and provide the capability to send and receive messages, possibly performing additional processing on structured attachments such as CCx visit summaries.

4.2.2 HISP
The “mail server” and supporting infrastructure service, such as DNS. Note that in this implementation the duties of the HISP can be shared amongst many players, because the NHIN Direct-specific encryption and validation steps can be performed with a fully-integrated “HISP”, or performed within an organization or even as a source/destination plug-in, in which the “backbone” functionality can be supported by existing ISPs.

Please see the discussion and diagrams in section 1.2 for more description of how our implementation deals more with “logical” HISPs rather than requiring a single monolithic stack.

4.2.3 NHIN Direct Message
An Internet Mail format email message as described by the Content Packaging workgroup.

4.2.4 NHIN Direct Source Edge Protocol
Any protocol used by a mail client to send a message to a mail server. This will often be SMTP(S), but also can include proprietary client/server protocols such as MAPI. We do not attempt to limit this flexibility in our implementation.

4.2.5 NHIN Direct Destination Edge Protocol
Any protocol used by a mail client to retrieve messages from a mail server. This will often be POP3(S) or IMAP4, but can also include proprietary client/server protocols such as MAPI. We do not attempt to limit this flexibility in our implementation.

4.2.6 NHIN Direct Address
An address compliant with the Addressing workgroup specification, in traditional Internet email endpoint@dotted.domain format. Used to identify senders and recipients in the headers of an NHIN Direct Message.

4.2.7 NHIN Direct HISP Address Directory
A DNS server responsible for translating NHIN Direct Addresses into (a) the IP Address of its associated HISP/mail server and (b) a public certificate for that address, using CERT records.

4.2.8 NHIN Direct Backbone Protocol
HISPs in our implementation will communicate with SMTP(S), as with any other normal Internet email. The decision whether to encrypt the SMTP channel is left open at this time; the presence of the NHIN Direct agent makes it really a trivial decision.
5 USER STORIES

5.1 Describe how each user story is supported by the implementation. ALL stories regardless of priority should be addressed; working code may be focused on the priority 1 items.

Because many of the user stories are similar, we will define a few repeatable elements up front, and then reference them in specific stories.

We make the following assumptions, about the various ways in which clinicians and consumer may gain access to NHIN-Direct. We have added on addition “machine interface” concept as well.

- **EHR + Integrated NHIN-Direct** – in this implementation, a fully-scoped EHR has integrated NHIN-Direct into the EHR’s “inbox” or “messaging center.” In so doing, the EHR has effectively become a HISP, and can transparently move messages to and from a provider’s inbox. Cerner’s implementation of this uses Java MailDir structures which are created by the SMTP server. The messages are moved from MailDirs to the provider’s inbox using proprietary “edge” code.
  - NHIN-Direct accounts are probably provisioned via an EHR Vendor process, and would be correlated with clinician’s login accounts so that a secondary NHIN-Direct login would not be required.

- **EHR + Standalone NHIN-Direct** – in this implementation, the EHR vendor has not integrated NHIN-Direct into the message center, so a standalone NHIN-Direct client is used. The client could either be a standard email client (see below) or could be a web-based custom email client (see below). Message movement from EHR to standalone client could likely be done using cut-and-paste (clipboard) or similar OS-specific tools.
  - NHIN-Direct accounts would be provisioned via a local HISP, since they are not provided by the EHR vendor.

- **Standard POP/IMAP email client** – in this implementation, a local HISP provides a POP3 or IMAP service. After provisioning the user, the user can select a standard desktop email client (Outlook, Thunderbird, Mobile Phone, etc.) to connect to the HISP-provided POP/IMAP service, supporting a TLS-only connection, and requiring that the user always use his assigned password to set up the account. Stronger authentication could be applied, if policy requires it.

- **Web-based NHIN-Direct client** – the local HISP may prefer to offer a simple web-based client as an alternative to using a desktop client. The web-based client would be accessible only via HTTPS and would require the user’s assigned password for login. (Other authentication, such as knowledge-based challenge questions could be required by policy.)

- **Mirth HL7 NHIN-Direct module** – several of the user stories involve machine origination or machine destination of a secure message. We propose to create a simple open source module that would allow for any standard interface engine to be attached to a specified NHIN-Direct account. We will reference here the Mirth engine, a well-known open source interface tool, but any commercial engine could be used as well. [http://www.mirthcorp.com/community/overview](http://www.mirthcorp.com/community/overview)
The Module would use common libraries (available in most languages) to access the SMTP service (for sending) and/or the “mailbox” files created by SMTP (for receiving.)

**Specific User Stories – details and notes**

1) Primary care provider refers patient to specialist
2) Primary care provider refers patient to hospital
3) Specialist sends summary care information (back) to the referring provider
4) Hospital sends discharge info to referring provider

All of these (1-4) are essentially the same. In each case, the sending and receiving providers have access to one of the NHIN-Direct channels defined above. For EHR-integrated users, CCD and other clinical documents can be attached using EHR tools. For standalone clients, either standard drag-and-drop or “upload file” tools can be used to attach any desired content. Since email clients already format attachments using MIME, no additional code is necessary. In the case of story #4, the hospital’s EHR may have an automated system for sending discharge summaries by calling into an augmented HL7 engine (see Mirth HL7 Module above) or by writing proprietary “edge” code to invoke SMTP services to send the message to local providers. Or the hospital could do this manually, by attaching the CCD (etc) to a simple mail client, as per above.

5) Lab sends lab results to an ordering provider

We assume the lab has access to a list of the NHIN-Direct addresses for common local ordering providers. Those email addresses could be built into the lab’s outbound interface engine such that instead of sending a HL7 message over a dedicated VPN, the lab’s interface engine adds the HL7 to a secure message and sends it via NHIN-Direct, using the hospital’s HISP. Or, something like the Mirth HL7 Module could be used in place of the hospital’s existing HL7 interface. The code to create a new message, add the HL7 as a MIME type, and invoke SMTP would be simple to create, using standard libraries available in many languages and scripting tools.

6) Transaction sender receives delivery receipt.

Our proposed implementation of SMTP will require that all destination HISPs must return a status message, to the sender, using the ordinary SMTP channel. We would require these status messages to be extended with a custom RFC8288-compatible coded header field which will be used to indicate the destination HISP’s response to the sender’s message.

These return status messages will be asynchronous, and will need to be stored by the sending system, using the “message id” field to assign status to the correct message.

7) Provider sends patient health information to the patient
8) Hospital sends patient health information to the patient
9) Provider sends clinical summary of an office visit to the patient
10) Hospital sends clinical summary at discharge to the patient
11) Provider sends reminder for preventative or follow-up care to the patient

These five user stories (7-11) are basically the same as stories 1-4 with the major exception that the receiver of the message is a patient, rather than another provider. This requires that the patient obtain an NHIN-Direct address. The most likely way for this to happen is via the patient’s chosen PHR. Both Cerner and Microsoft plan to offer NHIN-Direct accounts to all PHR users who ask for them. Cerner and Microsoft both plan to offer web-based NHIN-Direct clients to consumers, as described above. These web-based tools will be integrated within a web-based PHR. There is no theoretical reason why a HISP couldn’t chose to offer an NHIN-Direct account via a Standard email client to a consumer, but we don’t expect this to be a common case.

12) Primary provider sends patient immunization data to public health
13) Provider or hospital reports quality measures to CMS
14) Provider or hospital reports quality measures to State

These three user-stories (12-14) are very similar in that a provider is sending structured reportable data to a designated agency. The details provided for these user stories appear to assume that the provider has an EHR capable of NHIN-Direct access. We propose that these stories would best be addressed using the Mirth HL7 NHIN Direct Module, described above. In these cases, the EHR’s outbound interface engine would invoke the Mirth Direct Module and pass in the structured HL7 or PQRI data. In addition, the interface engine would be provided with the NHIN-Direct secure address for the designated agency.

It may make sense for each reporting agency to allocate a separate secure address for each organization that has contracted to send data. That way it will be easy for the agency to keep track of who has sent what.

On the destination side, each agency would likewise implement a Mirth NHIN Direct Module in their interface engine to dequeue the inbound messages from their chosen HISP. The Mirth engine would poll the SMTP mailbox and pull down messages as they arrive, decoding the structured MIME-encoded attachment for subsequent processing. Of course, this could all be done manually using a standard email client, but this would not be a very scalable approach.

Other User Stories TBD in the future

6 SECURITY & TRUST

6.1 Describe how each consensus requirement is supported by the implementation.

2.1 Use of x.509 Certificates.
The NHIN Direct Agent used in this implementation supports the use of x.509 certificates as described.

2.2 Certificate Authority Anchor Configuration.

2.3 Self-Signed Certificate Configuration.

2.4 Certificate Granularity

This implementation supports all of these points; whether organization or individual certificates or both are used is a matter of configuration.

2.5 Revocation.

The implementation uses online re-fetch of certificates to manage revocation.

2.6 Sender identification.

The implementation performs first an individual subject match, and falls back to an organization match before rejecting the message.

2.7 Encryption.

The implementation encrypts all messages using a public key of the recipient.

2.8 Ease of use.

There are three levels of service that support this requirement. In the “full service” HISP model, all certificate management can be delegated to the HISP. Users can configure their preferences by specifying domain names and addresses, and can transparently be granted certificates by their HISP.

The next level down asks the organization to obtain one certificate per trusted CA. Complexity here is on par with purchasing an SSL certificate for a web server, something thousands of organizations do on a regular basis.

Finally, the organization may take complete control of individual certificates, if they have developed procedures to manage this already that fit into their existing workflows.

7 COMPREHENSIVE HIE

7.1 Describe the mechanisms by which the implementation can interact with NHIN Exchange implementations.

To achieve universal addressability, every user of NHIN-Direct must be assigned an NHIN-Direct address, and must be accessible by the standard backbone protocol via a local edge protocol. These NHIN-Direct addresses may be provisioned to users who participate in NHIN-Exchange as well as to users who do not participate in NHIN Exchange. All legitimate NHIN-Direct addresses will use the designated backbone
protocol to communicate direct messages. Therefore, for the purposes of direct messaging, there should be little or no need to “bridge” from an NHIN-Direct message to an NHIN-Exchange XDR message.

We do expect that many NHIN-Exchange nodes will choose to host the SMTP servers necessary in order to provide NHIN-Direct services to their users, much the same way that a given ISP can provide email service on one port as well as HTTP service on another port. NHIN-Exchange nodes will likely want to reuse their existing user directories and authentication tools to control access to their NHIN-Direct services as well as to their NHIN-Exchange services. We do not foresee any technical problems with this approach.

If an NHIN Exchange node does identify a need to move data from an SMTP message into an NHIN-Exchange environment, the SMTP “step up” and “step down” code being developed by the IHE Concrete Implementation Group could be used as a gateway to transform the data as needed. There will be some messages for which “step up” is not possible; for example, a simple text/plain message will not have sufficient metadata to make the jump to NHIN-Exchange. We expect the gateway software to simply “bounce” the message back in this case, just as it would for other failure cases.

An open question is whether or not an NHIN-Direct message should be allowed to serve the purpose of a “Register and submit document set” function which would allow a Direct message to move data into an XDS Registry/Repository. This is an interesting possibility, but it is outside the scope of simple direct messaging. If desired, it could be handled by custom “edge” code as per the “step up” and “step down” examples developed by the IHE Concrete Implementation Group.

8 Other Workgroups

8.1 If the implementation deviates from requirements of the Addressing, Content Packaging or Individual Involvement workgroups, describe and explain the reasons here.

This implementation complies with all requirements from these working groups.

9 Cost and Complexity of Development

9.1 Describe what new code is required to create a complete implementation vs. what can be used off-the-shelf.

The vast majority of code in our implementation is off-the-shelf. The three key pieces that have been written new are:

- The NHIN Direct Agent, which uses standard crypto libraries underneath but manages the orchestration of the process and normalization of messages, etc.
• The per-package Glue code that inserts the agent into the email pipeline. This will be different for different SMTP servers; we hope to have one generic implementation that works as a standalone gateway as well.
• The certificate administration interface that allows a user to configure their certificates.

9.2 Describe library and required component availability across platforms and frameworks.
We have had encountered no problems implementing the code across three platforms so far using base libraries. We use standard development and test tools as well.

9.3 How does this approach integrate with existing mainstream healthcare applications? Describe cost/complexity of integration.
Many applications today already embed email functionality, especially email sending --- base library support for these operations is widespread and integration with our implementation will be exactly the same. Fetching mail is less supported in base libraries, but extremely well-supported in available open source and commercial components. MIME handling has been around for decades at this point and is well understood, low complexity and low cost.

9.4 Describe the anticipated departure or switching costs a vendor may incur with this approach (versus previously supported paradigms).
NOTE THIS CAPABILITY IS UP FOR REMOVAL BY THE CONCRETE IMPLEMENTATION WORKGROUP

10 Workflow / User Experience

10.1 Describe how a small (1-5 provider) practice with limited IT capability might deploy and service the implementation.
In the simplest case, a provider will contract with a Full Service HISP to provide NHIN Direct email service to the office. They will have a web-based administration tool for creating and managing addresses, just as they can do today for traditional email with an ISP. They can use the HISP-provided webmail client, or desktop clients of their choice (e.g., Outlook), or configure their EHR software to talk to the HISP directly.

Alternatively, a single practitioner might choose to download an NHIN Direct desktop plug-in, and use it with their existing email provider for NHIN Direct email.

10.2 Describe how a medium practice with an outpatient EMR might deploy and service the implementation.
A medium-sized practice might take the same Full Service HISP approach, and simply configure their EMR to talk directly to the NHIN Direct email services provided. The HISP might provide an
administration API that the EMR can connect to, to automatically provision and manage email accounts when new EMR accounts are created.

Alternatively, the practice might choose to install an NHIN Direct gateway and use a traditional ISP for email services, handling the provisioning themselves.

10.3 Describe how a large practice / hospital might deploy and service the implementation.
In addition to the same approach as in 10.2, a large practice might pull the entire HISP system (email servers, DNS servers, etc.) into its own IT infrastructure. Any organization that manages their own email systems today would be a candidate for this approach. An NHIN Direct agent plug-in for Exchange (using the “SMTP Agent” Exchange model) would be simple for a vendor to create.

10.4 Describe how a patient might participate in the implementation.
A typical patient will probably participate as part of a PHR account at a system such as Google Health or Microsoft HealthVault. These services can automatically provision NHIN Direct addresses for their users, and provide “inbox” functionality to receive messages and, if desired, send them as well. They will share their address with their providers in person as proof of ownership.

A patient might also use the same approach as in 10.1 to obtain NHIN Direct email service. It is unclear what certificate authority would grant certificates to these individual users, but if the demand exists there is no technical reason that one could not emerge.

10.5 Describe how a service provider might create and operate an implementation.
One of the best features of the SMTP implementation is that being a HISP is almost identical to being an ISP today. The service provider will have to install the NHIN Direct agent in their email pipeline, either using new email servers or existing ones. They will have to augment their administration interfaces to help manage certificates, and ensure that their existing DNS services can serve up certificates as well.

Note that beyond the agent there are zero new concepts for HISPs to learn – something that makes a big difference in complexity and manageability of the data center environment.

11 Miscellaneous

11.1 Metadata Handling
NOTE that responses here are really a result of conformance with the Content Packaging Workgroups recommendation to use RFC 5322 Internet Messages, not SMTP per se.

11.1.1 Describe what metadata inherently handled by the implementation
The only metadata required by the implementation is that specified as such by the Content Packaging workgroup: from, to, orig-date, message-id, in-reply-to, references, cc, and MIME-Version.
11.1.2 Explain how more complex metadata could be added to the implementation if it is desired
The Internet mail format used by this implementation can support arbitrary headers. Many email systems take advantage of this capability to transfer additional metadata, for example likelihood of a message being spam, or an “importance” level that can be reflected in the user interface.

In addition, per the Content Packaging workgroup, more complex metadata can easily be transmitted in XDM form as part of a message by adding it as a MIME part with type “application/xdm+zip”.

11.2 Content Handling
NOTE that responses here are really a result of conformance with the Content Packaging Workgroups recommendation to use RFC 5322 Internet Messages, not SMTP per se.

11.2.1 Describe how the implementation deals with unstructured text content.
Internet mail format can transmit any content describable with a MIME type, and we expect “text/plain” content to be the format for many useful human-readable exchanges.

11.2.2 Describe how the implementation deals with unstructured binary content (e.g., PDF).
Internet mail format supports arbitrary attachments; we expect the use of formats such as PDF, DOC, etc. will be widespread. No special processing is required by the implementation to deal with these attachments; all modern email clients will understand them automatically and how EHR integrations can do so is well-understood.

11.2.3 Describe how the implementation deals with structured content such as CCx.
See 11.2.2. Attachments can be in structured content types such as CCR and CCD. Clients that understand these formats can extract the attachments and perform further processing. For example, an EHR might show an interface for enabling a provider to integrate structured data from a CCx document directly into the patient chart.

11.3 Extensibility / Future-proofing

11.3.1 How well does the approach position NHIN for the phase-2 work in supporting queries, or a hybrid of queries and partial pushes (ie - push to HISP, then pull down specific addressed payload from HISP)?
NOTE THIS CAPABILITY IS UP FOR REMOVAL BY THE CONCRETE IMPLEMENTATION WORKGROUP

11.3.2 How would this implementation approach extensibility in general to support new use cases?
The Internet mail message payload format gives us great flexibility in terms of what content can be sent with a message. We expect that many new use cases will consists of attaching new structured content to messages in order to accomplish specific tasks; for example, a structured request for an appointment.

The agent approach makes it very easy to think about moving NHIN Direct messages over transports beyond SMTP as well. For example, we could imagine sending the same messages over instant messaging channels with no change in the way we manage the signature and encryption process.
12 ADDITIONAL CAPABILITIES

12.1 Describe any other relevant capabilities or use cases facilitated by this implementation

12.1.1 Message Threading
The references header in the Internet Mail Message can be used to create a threaded conversation. The header contains the message ids of all messages in the conversation. This enables a single, compact view of all responses related to a given conversation.

12.1.2 Direct Reply To
Using the RFC822 reply-to header, senders can specify a different destination for replies to a given message. This can be particularly useful in scenarios where providers want to initiate a follow-up message to a patient, with any replies being sent to a special inbox for appropriate triage.

12.1.3 “Out of Office”
Many email servers support the idea of automated “out of office” messages to be sent on first receipt of a message when the address holders is not available for an extended period of time.

12.1.4 Message Rules and Junk Email Filters
Email clients and some servers commonly support the ability to auto-sort incoming email based on complex content rules.

12.1.5 Virus Scanning
Many email servers will automatically scan inbound and outbound messages for viruses.

12.1.6 Enterprise Policy and other Content Filters
Many third party systems can plug into existing email servers to “watch” for inappropriate disclosure or other policy violations in email.

12.1.7 Foldering and Categorization
Most email clients have a capability for organization messages in folders, and many have the ability to “flag” items visually for follow-up in a certain period of time.

12.1.8 Address Book
Most email clients and servers support personal and group address books, making it simple to keep track of commonly-used addresses.

12.1.9 Read receipt
RFC2298: A MDN (Message Disposition Notification) typically includes a simple text message body that includes a declaration that the message was read and when. In addition, it includes a message part with headers that among other things indicate if the message was displayed, processed, dispatched, deleted, denied, or failed. The original-message-id header can be used to match the MDN to the original message during processing. Other interesting uses of this concept include notifying the sender if not
read within “x” days and in the case of multiple recipients, identifying the recipients who have reviewed the message and those who have not.

12.1.10 Multiple recipients
In addition to multiple primary recipients, a copy of a message can be sent to secondary recipients utilizing standard CC (carbon copy) and BCC (blind carbon copy) functions.

Also, Group Lists can be built to easily send an email to a defined group of recipients using a single alias.

12.1.11 Email aliases:
As defined in RCF5321, a SMTP server should support aliases. The destination HISP will replace the pseudo-mailbox address in the envelope with the real address and then deliver the RFC822 message unaltered. This can support many workflows including disposable email addresses, change in ownership of organizational mailboxes, distribution list type functionality, and much more.

12.1.12 Sub-addressing:
Most email clients support sub-addressing where the local part of an address is augmented by additional information. This has several uses including routing of messages. A “localaddress+tag@domain.com” scheme could be leveraged to automatically route to a folder related to that tag. For example, drsmith+12345@sunnyfamilypractice.com could be an email address that Dr. Smith gives to his patient with a MRN of 12345, with all messages being automatically routed to that patient’s folder. This is very similar in concept to a custom patient header, but instead can be defined real-time by the end user.