


18F

**CMS**

# **Contract Terminations Recommendations Report**



**12/2021**

## **18F Team**

Mike Gintz / [mike.gintz@gsa.gov](mailto:mike.gintz@gsa.gov)

Greg Walker / [greg.walker@gsa.gov](mailto:greg.walker@gsa.gov)

Randy Hart / [randy.hart@gsa.gov](mailto:randy.hart@gsa.gov)

# Table of contents

<b>Table of contents</b>	<b>2</b>
<b>Project overview</b>	<b>3</b>
Research	3
Problem statement	4
<b>Key findings</b>	<b>4</b>
<b>Recommendations</b>	<b>10</b>
For states	10
For vendors	18
For CMS	20
<b>Conclusion</b>	<b>23</b>
<b>Appendix</b>	<b>24</b>
About 18F	24
18F's De-Risking Guide	24

# Project overview

Medicaid system work depends upon successful partnership between states and digital vendors. Sometimes, though, those partnerships end earlier than expected.

Every contract termination indicates that something went critically wrong, and each ended relationship costs time and money. Terminating a contract is often the last resort after months or years of indicators that a contract will not deliver on intended outcomes. Even if a state and vendor terminate their working relationship without ending up in a legal battle over contractual deliverables, the resolution path from a termination is typically complex and costly for the government:

- The state must undertake a new procurement process to replace the old vendor and onboard a new vendor into a new and potentially problematic environment.
- The new vendor must then pick up the pieces from the last vendor, reverse-engineer how the partially-completed project was intended to function, and start building atop or around it—or start completely from scratch.
- The state, and CMS, must undergo a complicated and time-consuming process of identifying what portion of the expected value was truly delivered, in order to determine whether funds must be returned to CMS.

These steps add cost and effort to a project that was *already* in crisis.

A team from 18F, working closely with our partners in CMS’s Data and Systems Group, spent 12 weeks investigating recent contract terminations and near-terminations, interviewing states, state officers, and subject matter experts. This project—in some ways, arguably a large-scale [Agile retrospective](#)—sought to identify trends and deliver recommendations to reduce the likelihood, necessity, and impact of contract terminations.

---

## Research

Over the course of the project, we held 14 multi-round interviews with 28 participants, representing:

- 8 states
  - 5 that had recently terminated contracts,

- 1 that had narrowly avoided a termination, and
- 2 that had been described to us as having “effective” contract management/procurement practices
- 5 CMS state officers
- 1 non-profit (New England States Consortium Systems Organization (NESCSO)), and
- 1 vendor group (Private Sector Technology Group (PSTG))

We also reviewed contracts and background documents for all the state projects we spoke to state teams about.

After we concluded our research, we de-identified all interviews to preserve interviewees’ privacy, synthesized our findings, and formulated recommendations.

---

## Problem statement

Sometimes a contract termination is the right path forward, as opposed to slogging onward forever with no realistic hope for successful delivery. Projects that are headed in the wrong direction are not always recoverable. But terminations are still destructive, and should be used only as a last resort.

Our problem statement, therefore, was:

*How might states, vendors, and CMS make contract terminations less necessary, less frequent, and less destructive? What can teams do, in advance and during a project, to reduce the likelihood that it ends in termination?*

---

## Key findings

Our **key findings** represent the summarized results of our research:

- 1. CMS modularity requirements encourage COTS, but blurry lines and specific state requirements around modules often make a COTS approach impractical.**

The basic premise of a modular approach would seem to encourage the use of commercial off-the-shelf (COTS) software: a collection of interchangeable parts

that can be swapped in and out according to a state's needs or wants at any given time. If everyone has the same understanding about what the modules are and what they do, there's no reason to reinvent the wheel for every state.

The reality, however, is that the boundaries around what constitutes a "module" are fuzzy, and individual states have unique needs that make COTS impractical. Often, vendors describe their product as true, configurable COTS only for states (and sometimes the vendors themselves) to discover later that the product will require extensive customization to meet states' needs.

One example from our research related to electronic visit verification (EVV). States had different expectations of how that module would interact with their MMIS: some states wanted the EVV to generate MMIS case data and provide that directly to their existing MMIS, whereas others wanted the EVV to only provide visit data; their MMIS would create the case data. **Every state we talked to about EVV contracts believed they were buying a COTS product, but each of them ended up having to pay for extensive customizations.**

Another example concerned the provider services module. In this case, a vendor presented their product as COTS to two different states and then resisted making changes to it; this would be an appropriate response if the product was truly production-ready COTS, but the module didn't satisfy the states' identified requirements, to which the vendor had contractually agreed. The vendor had made different assumptions about the module than those states, and the states had different assumptions from each other. There was not, in fact, a single understanding of what the module was supposed to do.

We also heard about several instances where vendor sales teams had described and sold a product as COTS, but that same vendor's delivery teams were overwhelmed by the challenge of implementing it. These delivery teams were, according to their state partners, clearly trying their best and working hard (in some cases, unhealthily so), but were simply overmatched by the complexity of their task. In each case, key delivery team members burned out from the strain of the project and left the vendor company entirely, creating staffing gaps that made the project even less likely to succeed. State staff directed most of their frustration at the vendor sales teams, for promising results that their organization couldn't deliver. It's unclear to us whether the problem lay with the description of the product, the skill of the team, or both; either way, a lack of alignment between what was sold and what was deliverable resulted in months of missed

milestones, delays, and, eventually, termination—to say nothing of the effect on the public, who needed the services and systems that remained undelivered.

**2. The penalties CMS imposes upon states that miss CMS-mandated deadlines don't always encourage high quality outcomes. Sometimes, they even cause new problems.**

With the rollout of the Electronic Visit Verification (EVV) mandate, CMS imposed implementation deadlines and corresponding funding penalties for states in the hopes of incentivizing them to deliver promptly. Unfortunately, sometimes these deadlines appear to functionally trap states in projects and vendor relationships that are unhealthy or even outright failing. If a state feels that there's no way they can end their current vendor relationship, re-compete and re-award the project to a different vendor, re-start work, and still hit CMS deadlines, they may choose to continue onward with an active project that's going very badly in the hopes that they can salvage it; in this case they see a long shot as better than no shot.

In our investigation of contract terminations, these states, and their CMS state officers, identified these situations as instances where states “avoided terminations,” and initially, several were framed to us as success stories. **When we dug in more deeply, though, it didn't seem like “avoiding termination” was much of a win—these projects still had a high likelihood of failing; they were merely taking longer to do so.** If, eventually, a project like this does miss the deadlines and fail to deliver, it will have consumed significantly more time, resources, and funding en route to that failure than it would have if it had been terminated and re-procured sooner.

In the case of Electronic Visit Verification, the deadline for implementing the new module created additional constraints as states and vendors learned the new requirements simultaneously. Any vendors that managed to get a handle on how to implement the new requirements—or got a handle on selling that they could do so—ended up stretched across multiple states, negatively impacting their actual ability to deliver. In this case, it seemed like the deadlines artificially tightened an already small vendor market. And since the mandate was brand new, states were unable to find what they considered relevant past vendor performance to use as a reference during procurement.

**3. CMS’s current role is to help states navigate CMS-required processes (e.g., funding requests, certification), not to intervene to affect project outcomes.**

States we interviewed reported that CMS was a good partner for helping prepare and submit documentation, and in letting states know what funding options were available to them. For example, when a state was concerned about losing funding for future efforts if they canceled a poorly-performing contract, CMS offered assurances that the state wouldn’t face those penalties.

**However, when it comes to project outcomes, states felt unsupported by CMS.** One state told us that when they reported project problems to their state officer, they felt that although their concerns were heard and noted, they received no support in getting to a better outcome.

States said that when CMS knows a vendor is struggling in multiple states, it would be helpful for CMS to share that with states. They also wondered if having CMS involved in conversations with vendors would be helpful, because sometimes vendors didn’t seem to believe it when states relayed federal requirements to them.

**4. Vendors typically face minimal consequences from contract terminations.**

During contract performance, states can issue Corrective Action Plans (CAPs) and penalties based on vendor performance. Contracts can be structured such that vendors are only paid after certain deliverables are received or milestones hit. In certain circumstances, when states manage contracts rigorously and vendors fail to deliver as planned, vendors lose considerable money during the period of performance. However, vendors rarely face any additional or specific financial or reputational cost *because of a termination*.

Why is this? States usually terminate for *convenience* instead of for *cause*, which typically results in the state and the vendor parting ways “amicably,” with no outstanding damages assessed and with no money recovered by the state. Even when states believe there is adequate cause to justify terminating for cause and recover costs, they rarely do so because it opens them to litigation that—even if resolved in the state’s favor—can still drain staff time and morale, and destroy their ability to focus on moving on.

Unfortunately, when states terminate for convenience, part of the agreement usually prevents them from speaking publicly about the nature of how their project went off the rails. **Even in cases of egregious product misrepresentation or outright vendor incompetence, states that terminate for convenience are typically barred from proactively cautioning other states away from following in their footsteps.** State decision-makers are usually willing to accept this limitation in exchange for avoiding the possibility of litigation, even though it reduces their ability to spare other states the same fate they just experienced. In our research we found one vendor who was pursuing an active contract in one state *while in the midst of being terminated for its performance on the very same MES module in another state.* The two states were not in communication about this vendor's performance.

Multiple states said that CAPs would prevent a vendor from getting a future award. In practice, however, we noted that states hired vendors who had CAPs not only in other states but even in their own states. To the extent that these vendors suffered any reputational harm, it didn't seem to significantly impact their ability to get new contracts.

The upshot is that vendors generally have little reason to fear a termination. If a vendor is continuing to make money even though a project is going poorly, they might as well continue doing so as long as possible. The termination itself is merely when the money stops coming.

## **5. Rigorous up-front plans never capture the ever-shifting complexity of a software project.**

States we interviewed talked about lengthy RFPs containing hundreds of pages of requirements that took months or even years to produce. Despite the thorough planning and requirements-gathering, vendors often remained unclear about exactly what the state wanted, and the contracts still failed. Usually, all that planning came unraveled within the first few weeks to a few months of the vendor onboarding.

Four different states told us they saw problems within a month—one, within a single day. In some cases, vendor sales teams had made promises that their product or delivery teams couldn't meet. In other cases, vendors struggled to develop program management artifacts that stretched years into the future. One state was blunt that their vendor has misled them about what could be configured, resulting in more customization than anticipated.



In two particular cases, we heard of vendors who had misunderstood the state's needs. The vendors had proposed baseline products to customize for the state. However, because the contracts had detailed requirements, the project teams were locked into a particular path and could not adjust with new information. They couldn't switch baseline products or take a different route with customization, ultimately leading to contract termination.

In all of these cases, the states' work to create detailed plans and precise requirements did nothing to encourage successful outcomes; instead, those in-depth plans contributed to contract terminations by asking for the impossible upfront and then not providing a way to adapt as new information came to light.

**6. States often struggle to hire staff with the skills and experience that are necessary to successfully oversee MES projects. They often rely on their vendors not only to do the current work, but to plan future work.**

We know that, for a variety of reasons, states have a difficult time hiring and maintaining in-house software expertise. This forces them to depend upon external consultants and vendors not only to design and build the digital services they need, but to assess whether that work is being done properly. This is a clear conflict of interest.

A variety of mechanisms, such as IV&V, have arisen to address this issue, but most still fall short of providing a real-time perspective on how the work is going, leaning instead on retroactive reviews of work that was supposed to be completed, or in-advance assessment of work plans that may or may not come to fruition. This leaves state staff and decision-makers perpetually vacillating between two competing lenses: what already happened, and what is supposed to be happening next. Neither of these lenses provides clarity into what is happening *now*. Combine that lack of direct insight with states' lack of in-house software expertise, and it's easy to see why states often find themselves unable to steer work that they ostensibly control.

---

# Recommendations

We used our key findings to build a set of **recommendations** for states, vendors, and CMS itself which we believe will help reduce the necessity, frequency, and damaging impact of contract terminations.

## For states

1. **Approach software projects as if you are buying the services of a team of people, not as if you are buying a product.**

*Relevant key findings:* [1](#), [5](#)

Even if you're being sold a complete, off-the-shelf product, you're almost definitely buying a service: a team of people responsible for building or configuring software. A tool advertised as COTS will probably require configuration, and possibly installation and operations support. That support is the service you *really* need. Unless you are buying an existing software product that does not need any ongoing expert configuration, installation, or operations—for example, Microsoft Office—you are buying services.

Fully-custom software development is also a service. The whole reason you're building new software is because there is no existing product that works for your needs. **Avoid the temptation to define a product in the contract—instead, describe the problem you need to solve and the outcomes you want. Then, let vendors propose how they would approach solving those problems and delivering those outcomes.** Look for approaches that include usability/UX research, user-centered design, and iterative development. These indicate a team that can be flexible and adaptive as requirements evolve based on user feedback and other learnings.

One nice side effect of this approach is that it's often easier and less complicated to use—for everyone, including the procurement team—because your team no longer has to define every detail of how the system will work in advance. Because you are hiring a software development or support service, you are freer to adjust what you want them to work on as you learn more.

**a. Structure contracts around hours worked, not around deliverables.**

At first glance, structuring payments around hours worked may sound risky—but as long as you cap the total contract amount, it in fact reduces risk in a handful of ways. Paying for hours allows the vendor to staff up or down as project needs fluctuate so the project team can adapt as conditions change. And since you only pay for hours worked, if there are delays with staffing or a team member goes on vacation, you don't pay for that time. Finally, if you determine that the vendor's work is poor quality or they lack the skills to meet your needs, you can simply stop assigning them work, which is effectively the same as ending the contract but without the overhead.

Structuring your contracts around hours and materials actually used is also an effective up-front acknowledgement that you don't know exactly what you're going to need over the entire life of the contract and that you're open to change based on what the project team discovers. This is the most effective way to design and build software.

Additionally, if you plan to take this approach in your contract, you can publicize it in advance as part of your procurement process and increase your likelihood of finding a vendor that can deliver effectively with this model.

**b. Establish a standard definition/requirement of COTS that goes beyond merely whether the product requires customization, but also includes how the product will be supported in the future.**

If the product you're buying—or that a vendor is proposing—requires any code changes, it isn't COTS. But oftentimes, a product is pitched as being *configurable*, even though those configurations can themselves amount to equivalent effort to customization. Your definition and understanding of COTS should account for customization that appears to be configuration. It's not uncommon for configuration to require making changes in computer-readable text files or database tables. These are not changes that most program office staff can make on their own, so when planning a configured COTS solution, also plan for supporting that configuration in the future.

**If the original vendor is the only provider who can maintain your configurations, it barely matters whether or not the product is truly COTS; you're still locked into one vendor in perpetuity.** You may even need to be mindful of sole-source procurement rules that could force your system into an early retirement simply because you can no longer configure it.

Ask your vendor how configurations are made to their product so you can understand the level of effort it will take to make changes in the future. Can anyone besides them do this work? What happens when their software is updated: are necessary configuration changes handled automatically, or do they need to be manually fixed? These questions will help you make decisions about your long term support needs and determine whether a proposed tool is truly, functionally, pragmatically, COTS—or whether that label is just being used as a sales tactic.

**2. Look at modularity not as a goal itself, but as a mechanism for reducing risk and introducing balance into your MES, as well as increasing the speed of your delivery of service.**

*Relevant key findings: [1](#)*

Modularity is a means to an end, not a goal of its own. Systems that leverage modularity reduce long-term risk by making it easier to swap out individual pieces as needed; making it possible to work on pieces independently and concurrently; and making it easier to bring in multiple vendors to share the load. The goal is to build a system that is more resilient to change—both anticipated and unanticipated.

Modular systems can be resilient in a variety of ways. For example, if they're deployed in a service-oriented architecture (SOA), your system can still be significantly operational even when a module fails. If a module is no longer meeting your needs, you can replace it more easily because it is not entangled into a larger, monolithic system. If a vendor is underperforming, other vendors can continue their work without significant hindrance or burden.

**The goal of modularity is resilience and risk reduction, not “making modules.”** Don't create modules just for the sake of being modular. Think about modules through the lens of risk: what module definitions help you reduce the risks to your overall system and program mission? How can you break up your overall system in a way that makes it more resilient to change?

**a. Diversify vendors across your MES.**

One of the benefits of a modular approach is that your modules are largely independent and can therefore be built by different teams. As a result, you can reduce your dependency on any single vendor by using a variety of vendors across your MES. Having multiple vendors, and a better ability to replace them, gives you more leverage in controlling your MES. Conversely, allowing one vendor to support too many of your MES modules makes them a single point of failure, increasing your dependency upon them and making your MES *less* resilient.

**b. Develop modules that provide value to users; don't just replicate your existing system.**

We often see “modular strategies” that involve replacing a monolithic system with modules that merely replicate the current system’s structure. This approach is a lost opportunity to improve the system that you’re spending so much time modularizing.

Rather than focusing only on implementing a backend system to “modularize” the components, focus on building functionality that’s immediately valuable to users and that allows for interoperability. This will help you avoid spending too much time researching “the perfect” infrastructure and will instead let your modular ecosystem emerge as user needs are better understood.

**3. Collaborate closely with your vendor instead of relying on the contract as a project management tool. Prioritize adaptability and responsiveness over predictability.**

*Relevant key findings:* [1](#), [4](#), [5](#)

Contracts are necessary legal documents to protect you when you hire a vendor—and to protect the vendor, when they hire you as a client. They establish a framework for how you will work together; they are, essentially, relationship agreements.

Contracts, however, are terrible tools for managing software projects. Their language is typically dense and hard to navigate, is intended to be *difficult* to modify (as opposed to being responsive to changing needs), and is designed for the needs of a legal/procurement user, not a software project team member.

A good contract will set the parameters of *how* a state and a vendor will work together, but it does not have to explicitly define exactly *what* you will work on. **Instead, use the contract to broadly define the outcomes you want to achieve together, and then engage closely with the project team as they work to identify solutions.** Understand what they're doing, how they're doing it, and what obstacles and complexities they're discovering.

Successful collaboration is more than receiving report-backs or presentations every week or month; it's about being a participant in the project team and having a presence in meetings where work is being done. Even if you aren't already a software expert, you'll be surprised at how thoroughly you come to understand the important elements of the project.

On a healthy project, once the contract is signed, nobody ever has to talk about it again. This may seem far-fetched at first, but if you're meeting with your vendor multiple times a week and working closely together, your conversations will quickly become more specific, relevant, and useful than referring back to the contract.

**a. Issue draft RFPs and solicit public feedback from vendors to inform the final RFP.**

Letting vendors see your RFP before you officially start asking for proposals gives them a chance to ask questions to clarify what you're asking for and to test their assumptions. Doing it publicly allows all vendors to learn from each others' questions, reducing the workload on the procurement team. Getting these questions answered early helps refine the proposal, so everyone is clear on what you want, and cuts down on the need for negotiation after award.

**b. Don't accept proposal assumptions. Work through them prior to awarding.**

Proposal assumptions are a trap: they typically place an unrealistic burden of complete system knowledge on a state and allow a vendor to significantly renegotiate their responsibilities if anything the state thought was true ends up being inaccurate. They provide this opening even if those inaccuracies have no material impact on the work that needs to be done.

As noted in [Finding 5](#) above, you are virtually guaranteed to discover surprising things about your system while working on it; **you shouldn't allow your negotiated contract to depend upon a degree of knowledge that's impossible for you to have.** Instead, work with your vendor prior to award to identify which of the assumptions they want to include are *critical* to the progress of the project, and do the research into those elements of your system to answer them definitively—again, *before* award.

**c. Issue awards based on the terms in the RFP rather than negotiating final terms after award.**

The RFP should already describe your needs and requirements. Any deviations made after bids and evaluation should be met with a high degree of skepticism: changes requested by the state undermine the careful crafting that went into the RFP, and changes requested by the vendor raise questions about why they bid on something they believed needed changes even before the work started.

If you're using draft RFPs that can be tweaked based on vendor questions and feedback *during* the procurement, negotiating *after* award is an unnecessary time sink that starts the project off by undermining what was just settled upon. Instead, let vendors know that their bids are binding, and that, if accepted by the state, the proposal represents the contract. Now, once a vendor is selected, work can begin sooner rather than waiting additional months for the negotiation of terms that shouldn't be up for debate.

**d. Assume that change will be needed. Prioritize resiliency and flexibility in your contract, objectives, and goals. During the lifetime of the project, let the team adapt.**

A good contract is one that will deliver value. Change is inevitable as the project team learns more about the problems that need to be solved, so a good route towards delivering value is to promote flexibility and adaptability. The more detailed and granular a contract is, the more rigid it becomes, the harder it is to adapt to changing and new understandings of the problem, and the more likely it is that the contract—and the project—will fail to deliver value.

Again, a good way to introduce more flexibility into your contract is to focus the language on objectives and goals instead of rigid requirements. By leaving the fine details out of the contract, you allow the project team to flexibly and incrementally choose the best path towards your desired outcomes without needing a formal contract change.

#### **4. Reduce the complexity of your procurement process to help your team manage multiple projects simultaneously.**

*Relevant key findings:* [1](#), [4](#), [5](#), [6](#)

Managing contracts is already challenging, but in a modular MES, you may be managing multiple contracts at the same time, and it will likely be overwhelming if you don't simplify them. Fortunately, if you no longer plan to use your procurement documents as project management tools, you can probably simplify them. A good rule of thumb: if your procurement is longer than 75 pages, it's probably too long. It's too detailed, too specific, and too rigid. A procurement that complex is burdensome to create, burdensome to manage, and burdensome for vendors to read, understand, and respond to.

The work that happens once a vendor is onboard usually mirrors the way work happened during the planning process. If you want a vendor to be flexible, responsive to change, and to prioritize what's most important, you have to figure out how to nurture that type of behavior in your own practices.

##### **a. Look for ways to standardize your procurement documentation to increase reuse and reduce overhead.**

You likely have required language for every contract you write. You also probably have language that appears frequently. As you develop new RFPs to reduce complexity, identify the parts that will always be included. **In particular, the most significant difference between RFPs should be in a brief problem statement and in the statement of objectives.** Most of the rest of the document should become boilerplate.

Use consistent and standardized language in your procurement documents to enable you to release them more quickly. They'll be faster to write and approve, since only a small amount of text will be unique to each RFP. They'll also be easier for vendors to read and respond to, since once they've read and understood one RFP, all subsequent contracts will be the same, just with a different problem and scope.



**b. Involve procurement staff early in the process so they can help contract in ways that work for your project.**

Many procurement teams know how to wade through some of the bureaucratic barriers that might hamper your team—but they won't be of nearly as much assistance if they're brought on at the tail end of your procurement planning.

If you establish an early relationship with your procurement team, they'll be more open to working in different ways that could lead to more successful outcomes. Most procurement professionals are open to exploring creative ways to work within the rules while helping find the right market to attract vendors. **Your shared goal should be to streamline the procurement process and reduce the barrier of entry for vendors that specialize in delivery.** The procurement team will be able to help you figure out which of the terms and conditions that often scare off potential vendors might be unnecessary, so you can approach your project as a team effort and engage vendors transparently right from the start.

**5. Create and hire an enterprise technical strategy role to shepherd your MES.**

*Relevant key findings:* [1](#), [5](#), [6](#)

States, especially those that struggle to hire in-house software expertise, should prioritize hiring for the job roles that are *most* pivotal. One of these can be described as an “enterprise technical strategy” role.

**Medicaid program offices need someone who understands system architecture enough to be an active participant in planning the future state of the MES.** They should understand software, product, UX, Agile, and user-centered design well enough to know what “good” and “bad” technical work looks like. This person would be responsible for making sure no one vendor exerts an overly heavy influence on the development of the MES, so they must be empowered to push back on vendors.

This role must be accountable to the program office, not an IT or CIO office. However, they must also be taken seriously by those offices. They need to serve the MES specifically, not the whole state, and they need to be able to push back on statewide policies that interfere with a successful MES.

This role could be filled by a state employee or a contractor to the state, but if they are a contractor, they should not be employed by any of the contractors working on your MES, to avoid conflicts of interest.

A good analogy for this role is a general contractor for home renovations: they're the person who brings together all the other experts doing the work and the client. They aren't a plumber or an electrician, but they know enough about both fields to guide a client towards making good choices about how to approach their plumbing and electrical problems and needs. They know how to evaluate plumbers and electricians, so they can hire good ones, and they can review work to ensure its quality. They help a client understand, fundamentally, *what's possible* and they have a big-picture understanding of the overall project. Their job is to connect their big-picture understanding to the specific subject matter expertise that the subcontractors bring in, and to make sure that the subject matter experts are delivering properly. This means their job is mostly about communication, to reduce the imbalance of expertise often found in state/vendor relations.

---

## For vendors

### 1. **Align your sales teams and delivery teams.**

*Relevant key findings:* [1](#), [5](#)

Sales teams face pressure to win contracts, but it's bad for business to sell capabilities that do not actually exist. If you are selling true COTS, be sure the team responding to RFPs has a good understanding of what the product does, how it can be configured, and the level of support necessary to configure and maintain it. Don't sell what you don't have without consulting your product or delivery team first to make sure it's a reasonable change that makes sense for your product.

If the RFP calls for key personnel, make sure those people will be available to work on the contracted project. Swapping out personnel between bid and award can feel like a bait-and-switch. If anything goes wrong, it could lead to the state questioning whether they got your best people, and that can be demoralizing to your staff.

**2. Treat RFPs as the standard you will be evaluated against, both for selection and in contract execution—not just an invitation to submit a proposal.**

*Relevant key findings:* [1](#), [5](#)

When a state issues an RFP, a significant amount of work has gone into it. Whether that's in-depth requirements gathering, customer feedback sessions, or the legal and policy processes to get a document out the door, the RFP is not just a lot of meaningless legalese. States often have required language that must go into RFPs, and that language is binding.

- a. Don't bid on bad RFPs in the hopes you'll be able to right the project after you win. Provide feedback on the parts that don't make sense in advance. If you still see red flags in the final RFP, don't bid, and tell the state why.**

Instead, explain to the state what you think is problematic about their request first. Try to negotiate to a better place *before* you bid to save yourselves and your state partners a lot of headache.

**If you've tried to negotiate, but you still see serious red flags, don't bid.** For example, if you sell a COTS product but the RFP includes requirements that are specific to that state and your product doesn't support them, that's a good sign that you should skip that procurement. Another red flag might be a procurement with hundreds of rigidly-defined requirements and no mechanism for changing them. That kind of rigidity is almost certainly doomed to failure.

Instead of ignoring the warning signs, let the state know you won't be bidding and why. Working together, you may be able to help the state update its procurement practices in ways that make it easier for both of you to find success.

- b. Anticipate that state legal/policy requirements will be very rigid, and that your state partners may not be able to be flexible even if they want to.**

Assume that contract requirements are fixed unless the contract explicitly says otherwise. If, later on, you identify a need to shift focus or pivot and your state partners agree, they may not be able to do anything about it. Don't assume that they can be flexible. Remember that they are

government entities, and they are more tightly bound to the legal language in the contract than might always be true in private contracts.

---

## For CMS

### 1. **Establish a definition of COTS to help states identify when configuration crosses over the line and becomes customization.**

*Relevant key findings:* [1](#), [2](#), [3](#)

CMS often finds itself having funded projects that initially appeared to be based on configuration, but have since ended up requiring customization—UMOTS<sup>1</sup> wolves dressed like COTS sheep. [Like states](#), CMS could take responsibility for drafting and publicizing an official, usable definition of COTS for its own funding purposes, with the added bonus that a standardized definition at the CMS level would be useful for states and vendors, too.

At the very least, CMS should make absolutely certain that its preference for funding configuration-based projects doesn't directly or indirectly nudge states towards interpreting or describing projects as configuration when they're truly customization. CMS should also help states identify when configuration-based projects are in fact at risk of *becoming* dependent upon customization:

- When reviewing APDs, RFPs, and contracts, state officers can be looking for language that shows the state is thinking about the costs of configuration, both in the short-term and the long-term.
- If the state hasn't thought about what it takes to maintain configuration over time, state officers can prompt them to do so.

### 2. **When establishing certification requirements for new modules, define and encourage data schemas as well, to guide implementers towards creating shareable, interoperable, exportable content.**

*Relevant key findings:* [1](#), [2](#), [3](#)

If CMS, potentially in partnership with states and the vendor community, can offer data schemas or API definitions as it rolls out new modules, that could improve outcomes by standardizing the market and making it easier for states to move between vendors if desired. We're not talking about enterprise-scale

---

<sup>1</sup> Unrecognizably Modified "Off-The-Shelf"

schemas, only the shape/structure of the data going into or out of a new module. The schema for a given module could even disregard the data schemas for any other parts of the MES; translating between modules should be the job of the integration component.

Module-level data standards are unlikely to be met with resistance from states. Instead, they could help make their procurements easier and more successful. It would simplify their contracts because they could directly reference the data standard. A market of vendors implementing a well-defined data standard would also eliminate one of the barriers to switching modules.

With standard data schemas defined, CMS could potentially explore standardizing API schemas for new modules in the future. Standard APIs would strengthen the benefits of data schemas, and open new possibilities. One ambitious outcome could be a set of standardized automated tests that states and vendors could use to quickly validate the behavior of a module.

### 3. **Phase the rollout of nationwide mandates to give the newly created "industry" time to learn and respond.**

*Relevant key findings:* [1](#), [2](#), [4](#)

Requiring every state to implement a new IT system at the same time within a relatively tight deadline is dangerous. Not only are all the states trying to understand the new requirement (and likely to arrive at different understandings of it) but they are also competing for limited vendor resources, because vendors also need time, to understand and be able to satisfy the new requirements. But vendors are incentivized to act quickly to sell their services... even if they aren't yet capable of meeting states' needs, and are hoping they can figure it out. States are similarly incentivized to take whatever they can get in order to meet the deadline and not be stuck without a vendor. And without any available record of how a vendor successfully delivered against this particular mandate—since the mandate is brand new, and *nobody* has ever delivered in this way—states tend to end up using whichever vendors are the first to claim they can get the job done.

**To change the incentives, phase in mandates if possible.** This could take the form of incremental mandates, where smaller requirements are rolled out periodically instead of all at once. Or, perhaps, introduce mandates to subsets of states rather than all at the same time. For example, consider choosing 2 or 3 states as “pilots,” to give CMS, states, and vendors a chance to start exploring

the implementation details and uncovering unexpected challenges before everyone faces them.

**4. Find ways to encourage and enable states to share learnings about failed contracts.**

*Relevant key findings:* [2](#), [3](#), [4](#)

As we addressed [above](#), terminations for convenience are much more common than terminations for cause. But when terminating for convenience, states often agree not to discuss project details publicly, leaving the true story of what happened buried, and removing any opportunity to learn from the experience. It would be preferable if states could openly discuss with each other when a contract goes poorly.

CMS may be able to guide states towards terminating contracts in ways that allow states to share what went wrong and what they learned. For example, perhaps CMS could require terms in the contract that prohibit states from entering into non-disclosure agreements related to termination. CMS has significant influence over states and the vendor community and it should leverage that to create environments where states can share and learn from each other.

**5. Conduct a research effort with states to identify desirable ways CMS can productively intervene to help "right the ship" on projects that are going poorly.**

*Relevant key findings:* [3](#)

We heard from multiple states that they wanted more support from CMS with regard to managing their MES implementations and vendor relationships. In one case, a state noted that their vendor didn't seem to believe them about CMS requirements. That state wondered if being able to call CMS into the conversation could have given them more authority. Other states believed that CMS knows which active projects are going well and which are going poorly and should proactively caution states if they have reason to believe the state is about to embark on a problematic project or contract.

CMS should collaborate with states to figure out how to best help, especially when projects are already going poorly. Identify the important moments in the project lifecycle that would most benefit from CMS's intervention so that you know not just *what* to do, but *when*. Be open to feedback and iteration, learning

what works and what doesn't as you go. Consider continuing retrospective conversations like the ones conducted in this project.

---

## Conclusion

While this project focused on contract terminations, a broader theme emerged during our research: the complicated web of relationships between CMS, states, and MES vendors makes it very difficult to pin down where true accountability for MES success lies.

When a MES project goes badly—when a system launches and is riddled with bugs, or when an old system stays active far past end-of-life—who ends up suffering? The public, who struggle to access the benefits they need.

The collective partnership between CMS, states, and vendors is intended to ensure this doesn't happen, but the lines of accountability are blurred: states wonder if they can trust their vendors, state regulations and procurement practices make it hard for even well-meaning vendors to pivot mid-project and react to what they learn doing complex work, and CMS is distanced from, and not held responsible for, state MES project failures.

The consequences that do exist for these parties don't always seem to incentivize success. Our research uncovered places where teams do the exact opposite of what would seem likely to lead to well-functioning software and effectively-delivered services. But given the complexity of the work and the existing incentives pushing and pulling on each team, everybody in the MES ecosystem seems to be doing exactly what would make sense for them to do in their position. Nevertheless, the outcomes often aren't what we seek.

What could change that?

Answering that question fully goes far beyond the scope of this project. We do, though, hope that the recommendations in this report will help states, vendors, and CMS take meaningful steps towards simplifying some of the complexity, confusion, and misalignment that leads to contract terminations—and by doing so, help to reduce risk and increase the rate of success on MES projects.

---

# Appendix

## 1. About 18F

18F is a digital consultancy housed within the Technology Transformation Services in the General Services Administration. It helps government agencies deliver exceptional digital experiences by building their capacity to practice human centered design, shipping often, and deploying products in the open.

Learn more about 18F at <https://18f.gsa.gov>

## 2. 18F's De-Risking Guide

If you found this report useful, you might be interested in two special handbooks 18F wrote to help state and federal agencies deliver successful technology projects.

- [The State Software Budgeting Handbook](#)
- [The Federal Field Guide](#)