eyevinn technology

WHITEPAPER

Resilience & business continuity

Emergency Broadcast as a Subset of Software-Defined Production

Why the best emergency broadcast solution is a smaller version of the production platform you should be running anyway, activated in one hour and grown in stages as the situation develops.

EXECUTIVE SUMMARY

Every broadcaster with a public-service or continuity obligation faces the same question: how do we stay on air if the main facility becomes inaccessible? The traditional answer, a second facility with duplicated broadcast hardware, is expensive, ages badly, and depends on equipment and muscle memory that are rarely exercised.

This paper argues for a different model. When production is software-defined and cloud-native, an emergency broadcast capability is not a separate system at all: it is a **subset of a real production solution**: the same software components, deployed small, on commodity hardware, independent of every headquarters system. And because each function is software that can be switched on rather than hardware that must be installed, the capability **grows gradually during the emergency**: one operator and one PC put a channel on air within the first hour, and richer production functions are enabled as more people and more time become available.

THE PROBLEM WITH PARALLEL EMERGENCY SYSTEMS

An emergency broadcast capability has one non-negotiable property: it must work when nothing else does. That means **complete independence** from the main facility: its network, its equipment, and its corporate authentication. A solution that needs the headquarters identity provider to log in, or a specific edit suite to produce content, fails exactly when it is needed.

The classic response is to build a parallel system: dedicated emergency studios, duplicated playout chains, hardware kept in a cupboard. This creates three chronic problems. The equipment is always a generation behind the main plant. The people who would operate it under stress have never used it, because it is not part of daily work. And every krona spent on it delivers value only in a scenario everyone hopes never happens.

> ***A system that is only used in emergencies is a system nobody knows how to use in an emergency.***

THE SUBSET PRINCIPLE

Software-defined production changes the economics and the logic. When mixing, graphics, rundown, editing, asset management, and playout are software services running on standard IT infrastructure, an emergency capability does not need to be a different system. It can be a **minimal deployment profile of the real one**: the same components, the same interfaces, the same skills, scaled down to what one operator needs in hour one.

Designing the emergency solution as a subset delivers what the parallel system never could:

- **Familiarity under stress.** Operators use the same browser-based tools in an emergency that they (or their colleagues) use in daily production. Nothing must be learned at 03:00.
- **Always current.** The emergency profile is upgraded whenever the production platform is; it cannot drift into obsolescence in a cupboard.
- **Testable at near-zero cost.** Because deployment is automated (infrastructure-as-code), the full emergency environment can be spun up, exercised, and torn down as a routine drill.
- **Honest economics.** The investment builds and matures the broadcaster's real production platform; the emergency capability is a deployment profile of it, not a second budget line.

The reverse reading matters just as much: if an emergency channel can genuinely run as a small deployment of your production platform, that is strong evidence the platform itself is what it should be: software-based, cloud-native, built on IT standards, and free of hidden dependencies on specific rooms, hardware, or networks.

WHAT THE SUBSET MUST CONTAIN

Distilled from real broadcaster requirements, the emergency profile spans six functions, each one a standard component of a software-defined production platform:

- **Access & activation**: reachable from any standard PC with nothing preinstalled, authenticated independently of the corporate identity provider, and operational within one hour of activation.
- **Live production**: a software vision mixer for TV and radio automation for audio: live hosting, prerecorded playout, and remote guests joining by phone, conferencing tools, or contribution streams from a plain browser.
- **Graphics**: logo and crawl/ticker with editable templates; picture-in-picture for accessibility services such as visual interpretation.
- **Content flow**: ongoing file ingest (music, pre-produced programs) from the main facility for as long as it remains operational, and pre-production of short programs that can be stored, edited, and scheduled for rebroadcast.
- **Coordination**: a simple rundown with countdown timers and previews, and a built-in low-latency intercom connecting everyone involved.
- **Distribution**: output as standard multi-bitrate manifests and transport streams, ready for pickup by CDN, cable, and distribution partners over ordinary internet paths.

GRADUAL ENABLEMENT: THE ESCALATION LADDER

The second design principle follows from the first. Because every function is software, the emergency capability does not have to arrive all at once, and must not be designed as if it did. The realistic emergency scenario starts with **one person and one PC**, not a full crew. The solution should therefore be structured as stages, each one enabling more functions as more people and more time become available:

STAGE	WHEN	CAPABILITIES ENABLED	CREW
Stage 0	Before any emergency	Preparedness: automated cloud deployment (infrastructure-as-code), pre-imaged laptops, documented network setup, quick-start guides, and regular drills	Platform team
Stage 1	First hour	Essential broadcast: one operator starts a basic live TV or radio transmission from a standard PC; built-in camera and microphone are enough; distribution partners pick up the output	1 operator
Stage 2	First day	Enriched production: additional operators join; remote guests via phone, conferencing, or contribution streams; external cameras and microphones; logo and crawl graphics; intercom	2–5 people
Stage 3	Sustained operation	Structured output: rundown and scheduling with timers and previews; pre-production of short programs stored, edited, and rebroadcast; ongoing content ingest from the main facility while it remains reachable	Small team
Stage 4	Extended operation	Toward full production: media asset management, editing tools, logging, and script management; the profile grows until it approaches the real production platform it was carved from	As available

Two properties make the ladder work. Each stage is **sufficient on its own**: Stage 1 is a legitimate broadcast, not a placeholder. And each stage is **a strict superset of the previous one**: enabling Stage 2 never requires rebuilding or replacing what Stage 1 put on air.

DESIGN CONSEQUENCES

Taking the subset principle and the escalation ladder seriously leads to concrete architectural choices:

- **Browser-first, install-nothing.** Every operator function must be reachable from a standard web browser. Anything that requires installed software breaks the one-hour, any-PC promise.

- **Independent identity and access.** The emergency environment carries its own user accounts and access levels (view/use vs. control), defined in advance.
- **Cloud-agnostic and containerized.** Deployable to public cloud, private cloud, or on-premises, wherever is reachable in the scenario at hand, with no integration lock-in.
- **Everything as code.** The entire server-side environment deploys from scripts; laptops are restored from disk images; network configuration is documented and automated. Activation is a runbook, not a project.
- **Standard formats at every boundary.** Small, standard media files internally; standard manifests and transport streams outward. Standard formats are what let each stage build on the previous one, and what keep every component replaceable.

CONCLUSION

Emergency broadcast capability is often framed as an insurance product: a cost, sized by fear. Treated as a subset of a software-defined production platform, it becomes something better: a forcing function for the modernization the platform needs anyway, and proof that it worked. The broadcaster gets a continuity capability that one person can activate in an hour, that grows in stages to a full production operation, and that is exercised, current, and familiar, because it is not a parallel system at all. It is the real one, deployed small.

Eyevinn Technology is an independent consultancy of video-streaming and broadcast software specialists in Stockholm. We design and build software-defined, cloud-native production and distribution solutions for broadcasters across Europe, including emergency broadcast capabilities on exactly the model described here. Independent means vendor-neutral: we help you select and integrate the right components, never sell you boxes.

Eyevinn Technology, independent video-streaming and broadcast specialists, Stockholm eyevinn.se · osaas.io